Hans-Joachim Adam / Mathias Adam

SPS

Programmieren in Anweisungsliste nach IEC 61131-3 (Lösungsbuch)

1. August 2012

adamis

Vorwort

In diesem Buch erhalten Sie Lösungsvorschläge zu den Übungen des zugehörigen Lehrbuchs. Sie können dieses Lösungsbuch von der Webseite der Autoren[1] herunterladen.

Für die meisten Übungen sind Musterlösungen verfügbar: Jede Übung hat einen eindeutigen Namen, unter dem die Lösung zu finden ist.

Die Übungen zum Teil I, Kapitel 2 bis 4, Digitaltechnik, können mit dem „Digital-Simulator“ durchgeführt werden. Dieses Programm kann kostenlos aus dem Internet heruntergeladen werden.[2]

Die Lösungsvorschläge für die Übungen zum Teil II, SPS-Technik, sind für das SPS-Simulationsprogramm „PLC-lite“, welches Sie von der Webseite der Autoren herunterladen können. Dort werden auch direkt in „PLC-lite“ einsetzbare Lösungsvorschläge für die Übungen und laufend Updates und Ergänzungen zu „PLC-lite“ bereitgestellt.

Bühl, Juni 2012

Hans-Joachim Adam
Mathias Adam

[1] http://www.adamis.de/sps/

[2] http://sourceforge.net/projects/digisimulator/files/

Inhaltsverzeichnis

Teil I
Digitaltechnik

Im ersten Teil des Buches behandeln wir die Digitaltechnik. Dieser Abschnitt dient als Vorbereitung zur SPS-Programmierung, die im Teil II behandelt wird.

Kapitel 1
Grundlagen: Zahlensysteme, Dualzahlen und Codes

Übung 1.1 Zahlenwörter

	deutsch	english	francais	espagnol	italiano
0	Null	Zero	zéro	cero	zero
1	eins	one	un	uno	uno
2	zwei	two	deux	dos	due
3	drei	three	trois	tres	tre
4	vier	four	quattre	cuatro	quattro
5	fünf	five	cinq	cinco	cinque
6	sechs	six	six	seis	sei
7	sieben	seven	sept	siete	sette
8	acht	eight	huit	ocho	otto
9	neun	nine	neuf	nueve	nove
10	zehn	ten	dix	diez	dieci
11	elf	eleven	onze	once	ùndici
12	zwölf	twelve	douze	doce	dódici
13	dreizehn	thirteen	treize	trece	trédici
14	vierzehn	fourteen	quatorze	catorce	quattòrdici
15	fünfzehn	fifteen	quinze	quince	quìndici
16	sechzehn	sixteen	seize	dieciséis	sédici
17	siebzehn	seventeen	dix-sept	diecisiete	diciasette
18	achtzehn	eighteen	dix-huit	dieciocho	diciotto
19	neunzehn	nineteen	dix-neuf	diecinueve	dicianove
20	zwanzig	twenty	vingt	veinte	venti

Übung 1.2 Lösung siehe Abb. 1.1

Abb. 1.1 Bündelfelder zu den Übungen 1.2 und 1.3

100	10	1
②	①	⓪
1	4	4

Übung 1.3 Lösung siehe Abb. 1.1

Übung 1.4 Ergebnisse:
a) 412 b) 130413

Übung 1.5

32	16	8	4	2	1
⑤	④	③	②	①	⓪
1	0	1	0	1	1

Abb. 1.2 Zweierbündelung zu Übung 1.5

Übung 1.6 Ergebnisse:

a. 011 = 3
b. 10101010 = 170
c. 111 = 7
d. 10010010 = 146
e. 100 = 4
f. 01010101 = 85
g. 10000 = 16
h. 10100 = 20
i. 110 = 6
j. 10000000 = 128
k. 1111 = 15
l. 10000101 = 133

Übung 1.7 Ergebnisse:

a. 3 = 11
b. 9 = 1001
c. 15 = 1111
d. 20 = 1 0100
e. 31 = 1 1111
f. 45 = 10 1101
g. 99 = 110 0011
h. 189 = 1011 1101
i. 200 = 1100 1000
j. 266 = 1 0000 1010
k. 278 = 1 0001 0110
l. 311 = 1 0011 0111
m. 499 = 1 1111 0011
n. 556 = 10 0000 1110

Übung 1.8 Ergebnisse:

a. 110 = 272 b. CD = 205 c. 1234 = 4660
d. EF6A = 61290 e. 3456 = 13398 f. 10F2 = 4338
g. 109 = 265 h. AE29 = 44585 i. 87E4 = 34788
j. 6E5B = 28251 k. AFFE = 45054

Übung 1.9 Ergebnisse:

a. 9 = 9 b. 45 = 2D c. 99 = 63
d. 580 = 244 e. 910 = 78E f. 3030 = 6D6
g. 6550 = 1996 h. 8750 = 222E i. 11111 = 2B67
j. 60000 = EA60

Übung 1.10 Ergebnisse:

a. 011 = 3 b. 10101010 = AA c. 111 = 7
d. 10010010 = 92 e. 100 = 4 f. 01010101 = 55
g. 10000 = 10 h. 10100 = 14 i. 110 = 6
j. 10000000 = 80 k. 1111 = 0F l. 10000101 = 85

Übung 1.11 Ergebnisse:

Dezimal -> HEX: siehe Tabelle in Übung 1.9.
Dezimal -> BCD:

a. 9 = 1001 b. 45 = 10 1101
c. 99 = 110 0011 d. 580 = 10 0100 0100
e. 111 1000 1110 = 78E f. 3030 = 110 1101 0110
g. 6550 = 1 1001 1001 0110 h. 8750 = 10 0010 0010 1110
i. 11111 = 10 1011 0110 0111 j. 60000 = 1111 0100 0110 0000

Übung 1.12

Zeichen	Bitfolge	Dez	Hex	**Zeichen**	Bitfolge	Dez	Hex
A	0100 0001	65	41	**0**	0011 0000	48	30
B	0100 0010	66	42	**1**	0011 0001	49	31
C	0100 0011	67	43	**2**	0011 0010	50	32
a	0110 0001	97	61	**8**	0011 1000	56	38
b	0110 0010	98	62	**9**	0011 1001	57	39
c	0110 0011	99	63	**(**	0010 1000	40	28
.	0010 1110	46	2E	**)**	0010 1001	41	29

Kapitel 2
Logische Funktionen und Boolesche Algebra

Übung 2.1 (EQUAL21) [1]

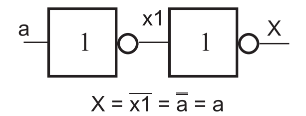


Abb. 2.1 Lösung zu Übung 2.1

Übung 2.2 (AND21)
Keine Musterlösung.

Übung 2.3 (HEAT21)

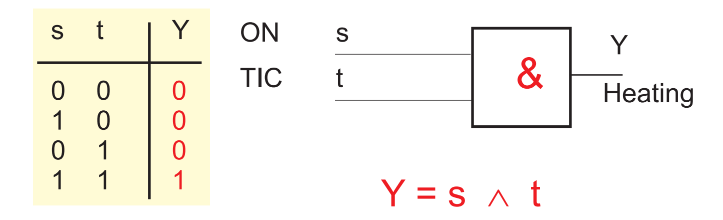


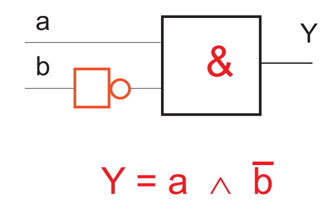
Abb. 2.2 Lösung zu Übung 2.3

[1] Mit dem „Digitalsimulator" (http://www.draw2d.org/digitalsimulator/about) wurden Musterlösungen erarbeitet, die Sie von der Webseite der Autoren herunterladen können.

Übung 2.4 (MIXER21)
Keine Musterlösung.

Übung 2.5 (ALARM21)

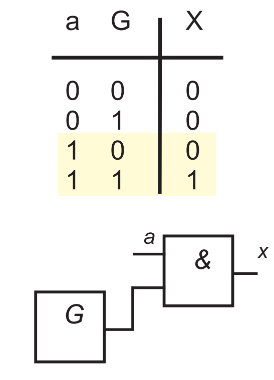
a	b	$\overline{b}$	y
0	0	1	0
1	0	1	1
0	1	0	0
1	1	0	0



$Y = a \wedge \overline{b}$

Abb. 2.3 Lösung zu Übung 2.5

Übung 2.6 (FLASH21)



a	G	X
0	0	0
0	1	0
1	0	0
1	1	1

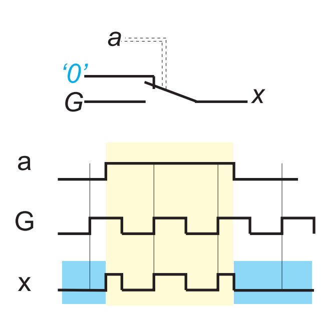


Abb. 2.4 Lösung zu Übung 2.6

Übung 2.7 (FLASH22)
Lösung siehe Abb. 2.5.

Übung 2.8 (NAND21)
Keine Musterlösung.

Übung 2.9 (NOR21)
Keine Musterlösung.

Übung 2.10 (LOGIK21)

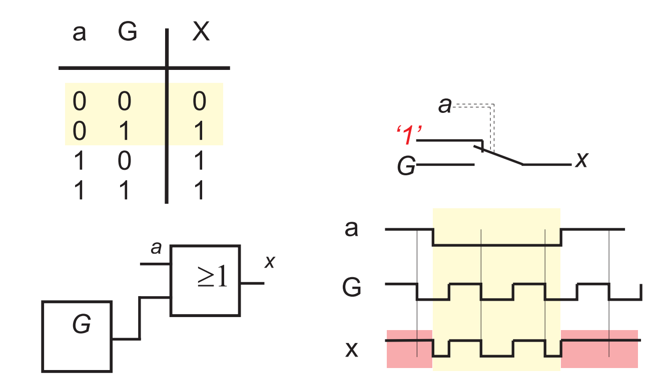


Abb. 2.5 Lösung zu Übung 2.7

Funktionstabellen, Logikpläne und Funktionsgleichungen zu Übung 2.10:

Funktion 1

a	b	x
0	0	0
1	0	0
0	1	0
1	1	1

$x = a \wedge b$

Funktion 2

a	b	x
0	0	0
1	0	1
0	1	0
1	1	0

$x = a \wedge \overline{b}$

Funktion 3

a	b	x
0	0	0
1	0	0
0	1	1
1	1	0

$x = \overline{a} \wedge b$

Funktion 4

a	b	x
0	0	1
1	0	0
0	1	0
1	1	0

$x = \overline{a} \wedge \overline{b}$

Funktion 5

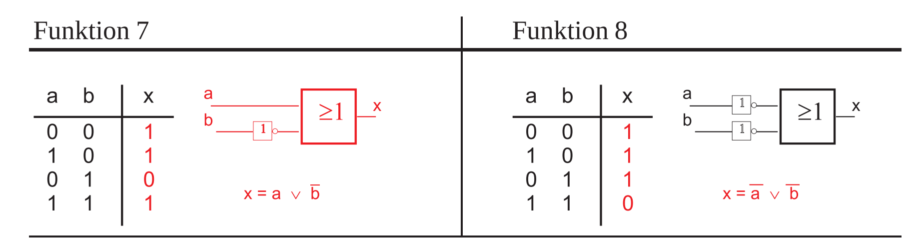
a	b	x
0	0	0
1	0	1
0	1	1
1	1	1

$x = a \vee b$

Funktion 6

a	b	x
0	0	1
1	0	0
0	1	1
1	1	1

$x = \overline{a} \vee b$



Funktion 7

a	b	x
0	0	1
1	0	1
0	1	0
1	1	1

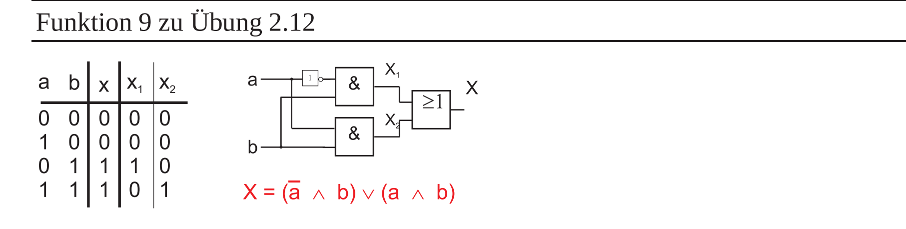
Funktion 8

a	b	x
0	0	1
1	0	1
0	1	1
1	1	0

Übung 2.11 (LOGIK22)
Keine Musterlösung.

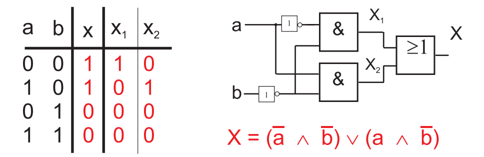
Übung 2.12 (LOGIK23)

Funktion 9 zu Übung 2.12



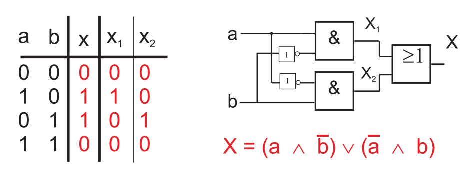
a	b	x	x_1	x_2
0	0	0	0	0
1	0	0	0	0
0	1	1	1	0
1	1	1	0	1

Funktion 10 zu Übung 2.12



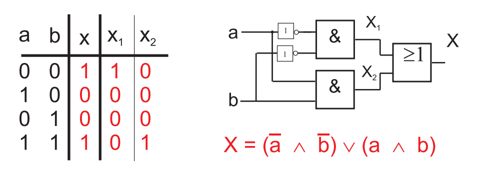
a	b	x	x_1	x_2
0	0	1	1	0
1	0	1	0	1
0	1	0	0	0
1	1	0	0	0

Funktion 11 zu Übung 2.12



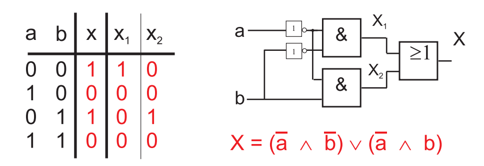
a	b	x	x_1	x_2
0	0	0	0	0
1	0	1	1	0
0	1	1	0	1
1	1	0	0	0

Funktion 12 zu Übung 2.12



a	b	x	x_1	x_2
0	0	1	1	0
1	0	0	0	0
0	1	0	0	0
1	1	1	0	1

Funktion 13 zu Übung 2.12



a	b	x	x_1	x_2
0	0	1	1	0
1	0	0	0	0
0	1	1	0	1
1	1	0	0	0

$X = (\overline{a} \wedge \overline{b}) \vee (\overline{a} \wedge b)$

Funktion 14 zu Übung 2.12

a	b	x	x_1	x_2
0	0	0	0	0
1	0	1	1	0
0	1	0	0	0
1	1	1	0	1

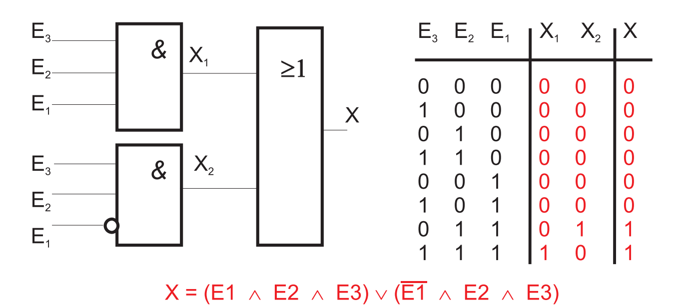
$X = (a \wedge \overline{b}) \vee (a \wedge b)$

Übung 2.13 (SWITCH21)

a	b	Y
0	0	0
1	0	1
0	1	1
1	1	0

Abb. 2.6 Lösung zu Übung 2.13

Übung 2.14 (MINTERM21)

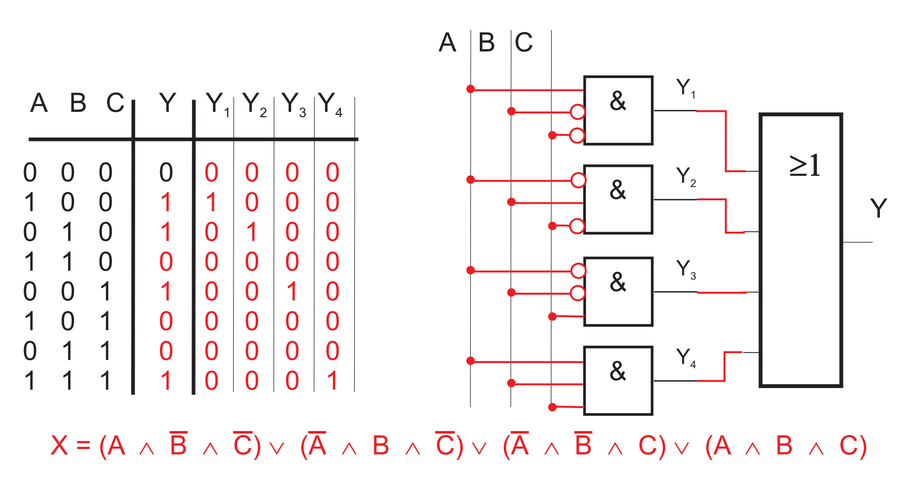


E_3	E_2	E_1	X_1	X_2	X
0	0	0	0	0	0
1	0	0	0	0	0
0	1	0	0	0	0
1	1	0	0	0	0
0	0	1	0	0	0
1	0	1	0	0	0
0	1	1	0	1	1
1	1	1	1	0	1

$X = (E1 \wedge E2 \wedge E3) \vee (\overline{E1} \wedge E2 \wedge E3)$

Abb. 2.7 Lösung zu Übung 2.14

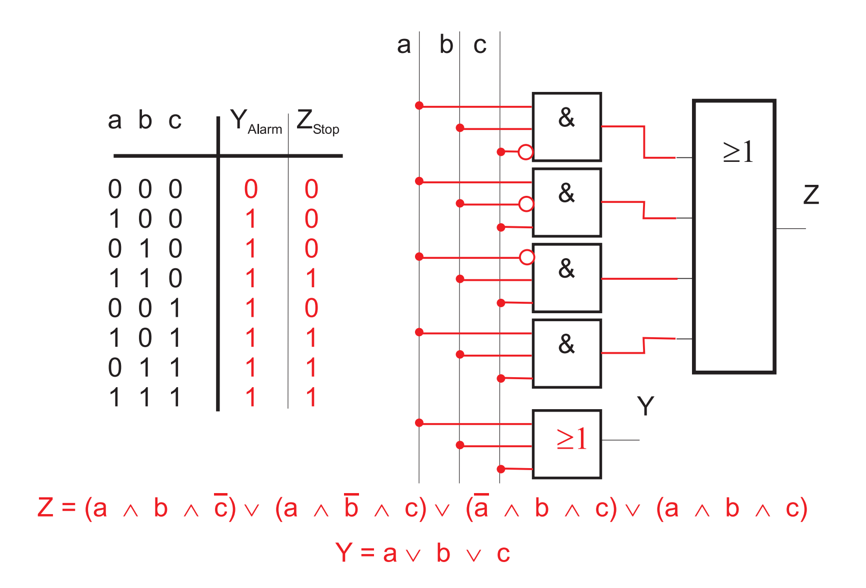
Übung 2.15 (SWITCH22)



A	B	C	Y	Y_1	Y_2	Y_3	Y_4
0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
0	1	0	1	0	1	0	0
1	1	0	0	0	0	0	0
0	0	1	1	0	0	1	0
1	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0
1	1	1	1	0	0	0	1

Abb. 2.8 Lösung zu Übung 2.15

Übung 2.16 (ALARM22)



a	b	c	Y_{Alarm}	Z_{Stop}
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	1	1
0	0	1	1	0
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

Abb. 2.9 Lösung zu Übung 2.16

Übung 2.17 (LOGIK24)

a	b	c	Y_1	$a \wedge b$	$a \wedge c$	$b \wedge c$	Y_2
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
1	1	0	1	1	0	0	1
0	0	1	0	0	0	0	0
1	0	1	1	0	1	0	1
0	1	1	1	0	0	1	1
1	1	1	1	1	1	1	1

Abb. 2.10 Lösung zu Übung 2.17

Übung 2.18 (LOGIK25)

a	b	$\overline{a} \wedge b$	$a \wedge \overline{b}$	X	$a \vee b$	$\overline{a} \vee \overline{b}$	Y
0	0	0	0	0	0	1	0
0	1	1	0	1	1	1	1
1	0	0	1	1	1	1	1
1	1	0	0	0	1	0	0

Abb. 2.11 Lösung zu Übung 2.18

Übung 2.19 (LOGIK26)
Lösung siehe Abb. 2.12.

S	R	Ya	$\overline{R} \wedge Ya$	Y	$\overline{R \vee \overline{Ya}}$
0	0	0	0	0	0
1	0	0	0	1	0
0	1	0	0	0	0
1	1	0	0	1	0
0	0	1	1	1	1
1	0	1	1	1	1
0	1	1	0	0	0
1	1	1	0	1	0

Abb. 2.12 Lösung zu Übung 2.19

Kapitel 3
Speicherglieder

Übung 3.1 (RSFF31)
Keine Musterlösung.

Übung 3.2 (ALARM31)

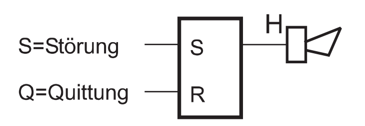
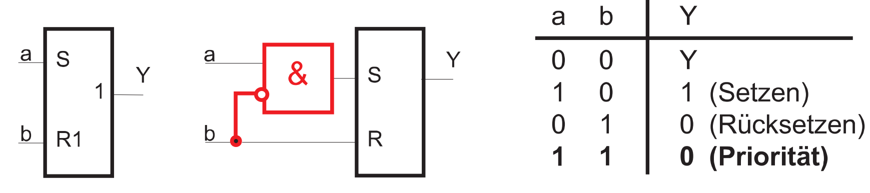


Abb. 3.1 Alarmschaltung1 zu Übung 3.2

Übung 3.3 (RSFF32)
Keine Musterlösung.

Übung 3.4 (RSFF33)



a	b	Y
0	0	Y
1	0	1 (Setzen)
0	1	0 (Rücksetzen)
1	**1**	**0 (Priorität)**

Abb. 3.2 Lösung zu Übung 3.4

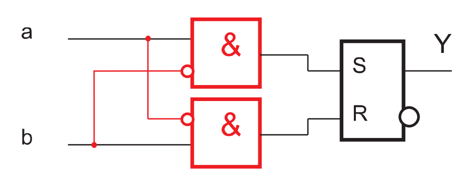


Abb. 3.3 Lösung zu Übung 3.5

Übung 3.5 (RSFF34)
Lösung siehe Abb. 3.3

Übung 3.6 (LIFT31)
Lösung siehe Abb. 3.4

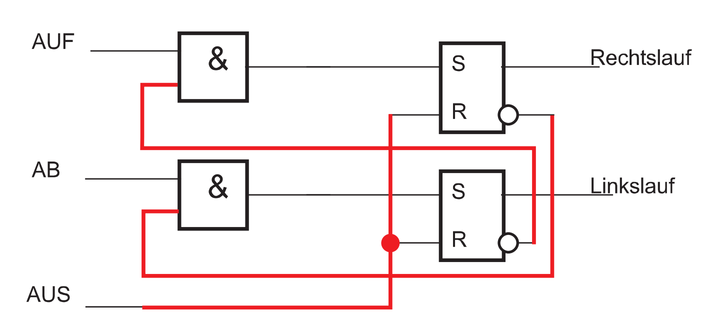


Abb. 3.4 Lösung zu Übung 3.6

```
Setzbedingungen für Rechtslauf:
    AUF und nicht Linkslauf
Setzbedingungen für Linkslauf:
    AB und nicht Rechtslauf
Rücksetzbedingungen:
    AUS
Aktionen:
    FF "Rechtslauf" gesetzt  -> Motor aufwärts
    FF "Linkslauf" gesetzt   -> Motor abwärts
```

Übung 3.7 (LIFT32)
Lösung siehe Abb. 3.5

Übung 3.8 (LIFT33)
Lösung siehe Abb. 3.6

Übung 3.9 (RSFF35)
Lösung siehe Abb. 3.7

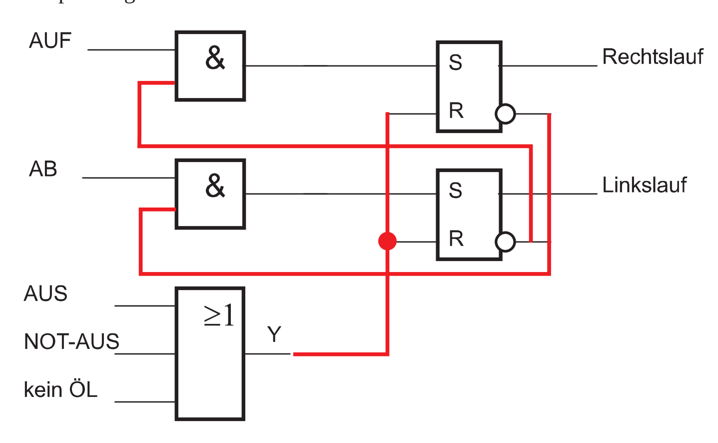


Abb. 3.5 Lösung zu Übung 3.7

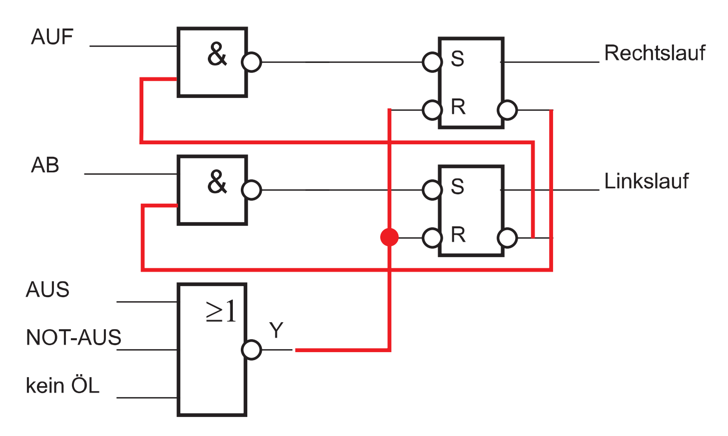
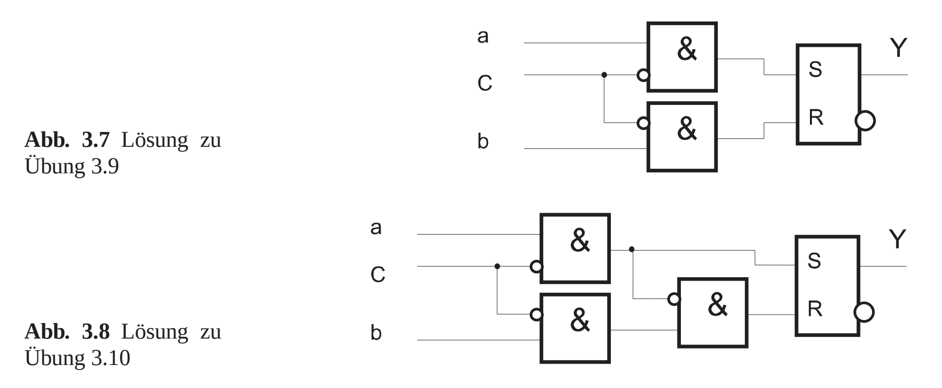


Abb. 3.6 Lösung zu Übung 3.8

Abb. 3.7 Lösung zu Übung 3.9

Abb. 3.8 Lösung zu Übung 3.10



Übung 3.10 (RSFF36)
Lösung siehe Abb. 3.8

Übung 3.11 (RSFF37)

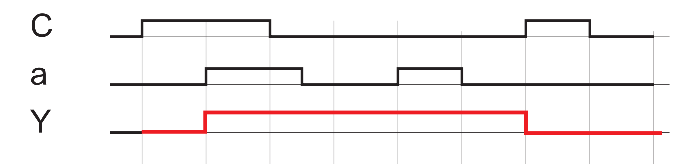
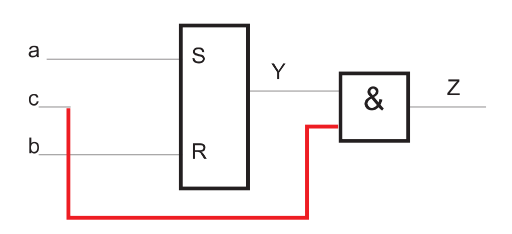


Abb. 3.9 Lösung zu Übung 3.11

Übung 3.12 (RSFF38)
Lösung siehe Abb. 3.10



a	b	c	Y	Z
0	0	0	Y	0
1	0	0	1	0
0	1	0	0	0
1	1	0	-	0
0	0	1	Y	Y
1	0	1	1	1
0	1	1	0	0
1	1	1	-	-

Abb. 3.10 Lösung zu Übung 3.12

Übung 3.13 (RSFF39)
Lösung siehe Abb. 3.11

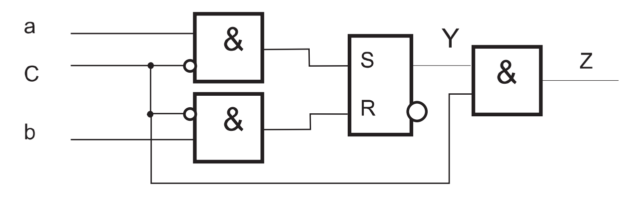


Abb. 3.11 Lösung zu Übung 3.13

Übung 3.14 (ALARM32)
Lösung siehe Abb. 3.12

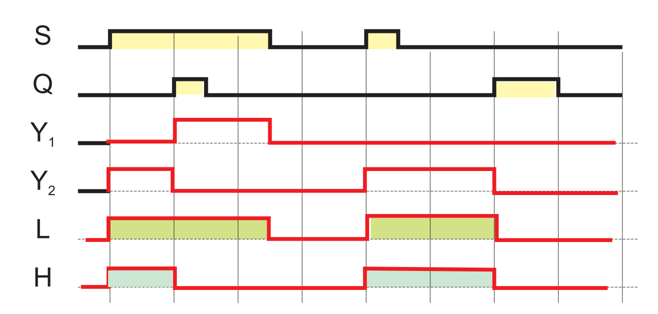


Abb. 3.12 Lösung zu Übung 3.14

Übung 3.15 (TANK31)
Lösung siehe Abb. 3.13

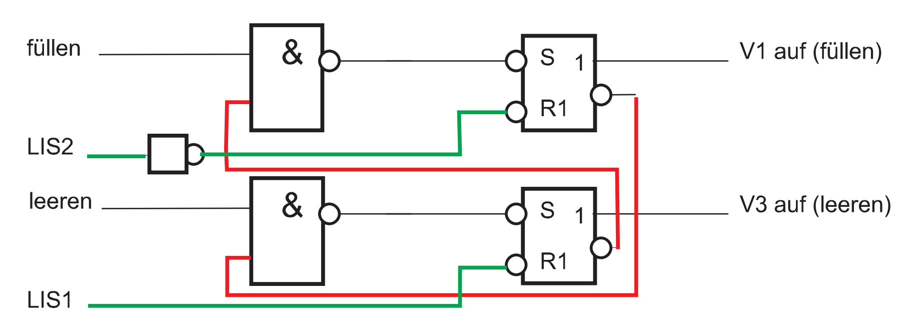


Abb. 3.13 Lösung zu Übung 3.15

Übung 3.16 (TANK32)
Lösung siehe Abb. 3.14

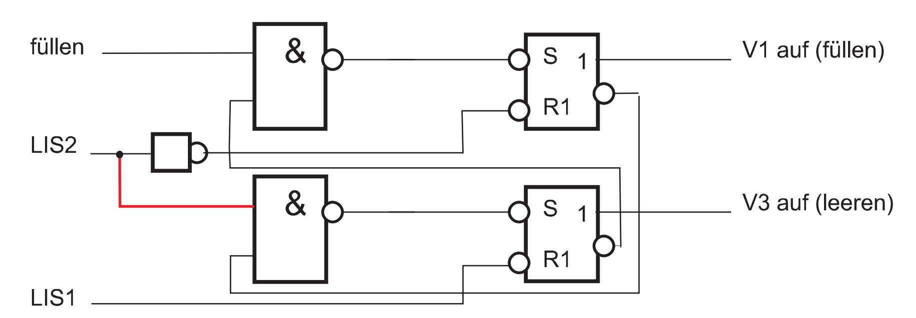


Abb. 3.14 Lösung zu Übung 3.16

Übung 3.17 (ALARM33)
Lösung siehe Abb. 3.15

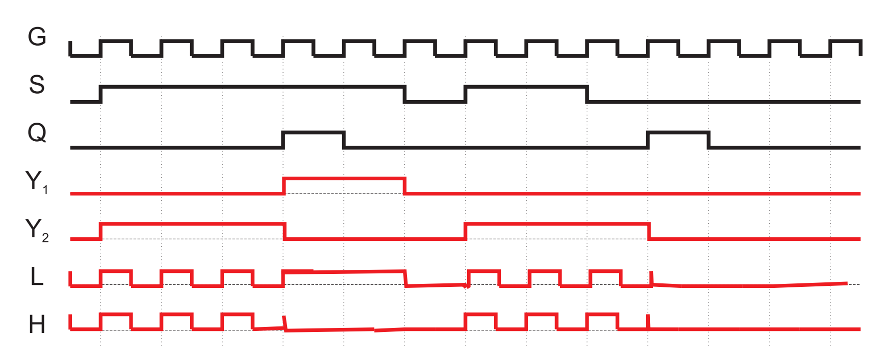


Abb. 3.15 Lösung zu Übung 3.17

Kapitel 4
Dynamische Speicherglieder und Zähler

Übung 4.1 (RSFF41)
Lösung siehe Abb. 4.1.

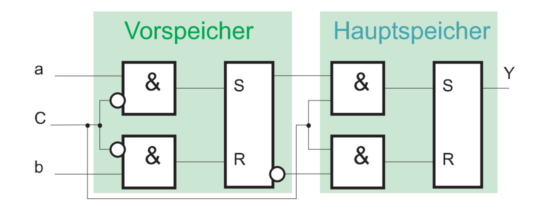


Abb. 4.1 Lösung zu Übung 4.1

Übung 4.2 (RSFF42)
Keine Musterlösung.

Übung 4.3 (RSFF43)
Lösung siehe Abb. 4.2

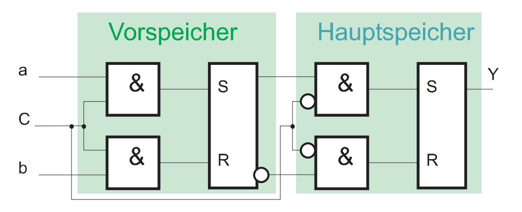
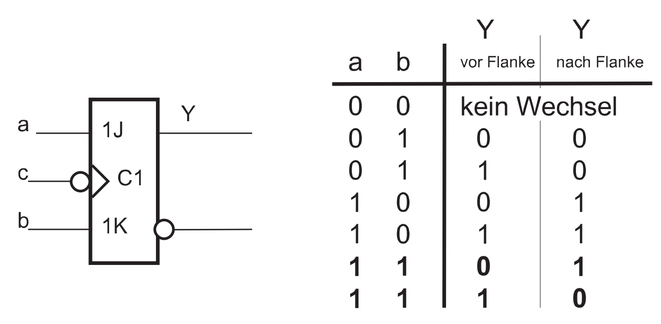


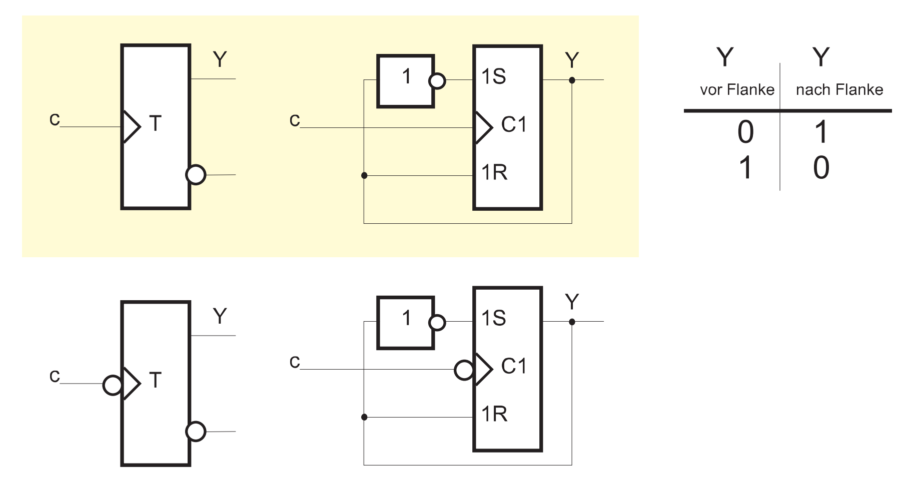
Abb. 4.2 Lösung zu Übung 4.3

Übung 4.4 (JKFF41)
Lösung siehe Abb. 4.3



a	b	Y vor Flanke	Y nach Flanke
0	0	kein Wechsel	
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	**1**	**0**	**1**
1	**1**	**1**	**0**

Abb. 4.3 Lösung zu Übung 4.4



Y vor Flanke	Y nach Flanke
0	1
1	0

Abb. 4.4 Lösung zu Übung 4.5

Übung 4.5 (TFF41)
Lösung siehe Abb. 4.4

Übung 4.6 (TANK41)
Lösung siehe Abb. 4.5
Wird an Stelle des T-Flip-Flop ein „gewöhnliches“ RS-Flip-Flop verwendet, dann sind zwei getrennte Taster für ’Start’ und Stopp’ oder ein Taster mit Umschaltkontakt erforderlich. Mit dem T-Flip-Flop genügt ein einfacher Taster und ein einfacher Tastendruck.

Übung 4.7 (COUNT41)
Siehe auch Abb. 4.6.
Die negative Flanke bringt das erste FlipFlop zum kippen, der Ausgang 2^0 wird zu ’0’. Dadurch entsteht eine negative Flanke, die das zweite FlipFlop kippt. Dies geschieht aber erst mit einer zeitlichen Verzögerung. Die Folge: kurzzeitig erscheint

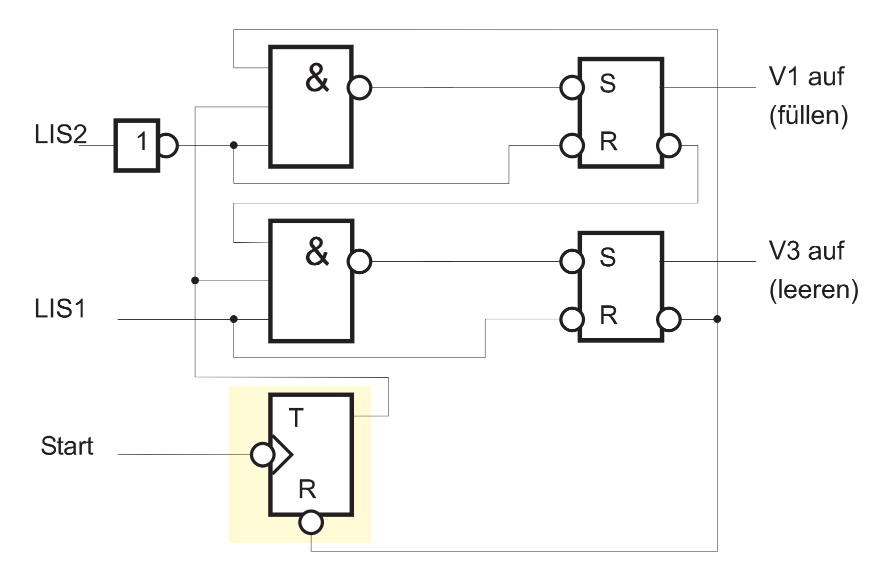


Abb. 4.5 Schaltung zu Übung 4.6

die falsche Zahl 1110. Das Kippen setzt sich fort, so dass nacheinander die weiteren falschen Zahlen 1100 und 1000 erscheinen bis erst zum Schluss sich das richtige Ergebnis 0000 einstellt.

```
Dez = HEX  =  8   4   2   1
...
14    E       1   1   1   0
15    F       1   1   1   1
0     0       0   0   0   0
1     1       0   0   0   1
...
```

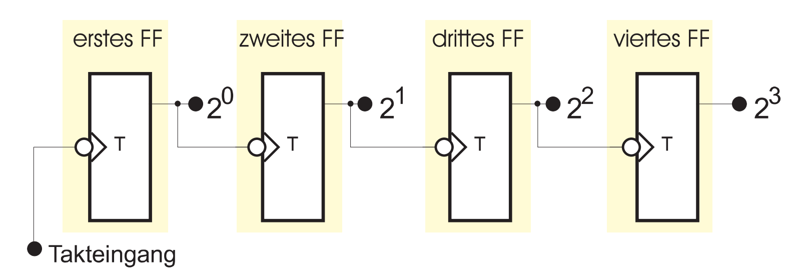


Abb. 4.6 Lösung zu Übung 4.7

Übung 4.8 (COUNT42)
Keine Musterlösung.

Übung 4.9 (COUNT43)
Lösung siehe Abb. 4.7.

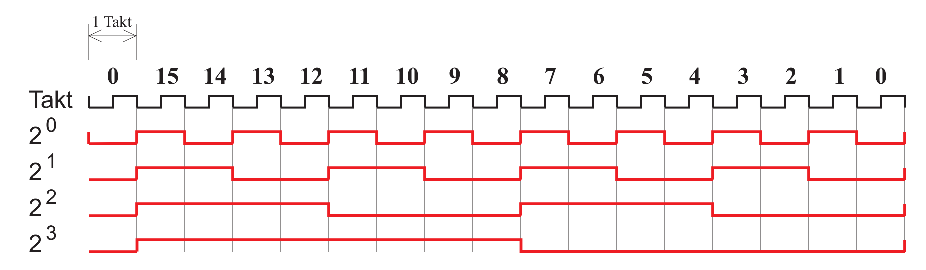


Abb. 4.7 Zeitdiagramm für 4-bit Zähler

Übung 4.10 (COUNT44)
Lösung siehe Abb. 4.8.

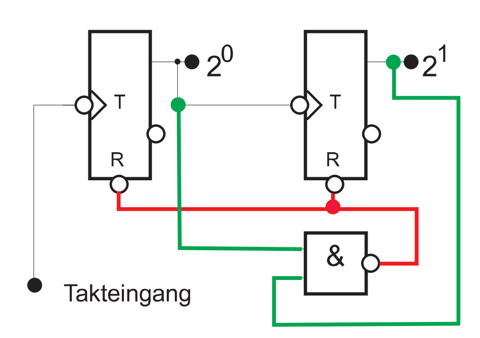


Abb. 4.8 Lösung zu Übung 4.10

Übung 4.11 (COUNT45)
Lösung siehe Abb. 4.9.

Übung 4.12 (COUNT46)
Lösung siehe Abb. 4.10

Übung 4.13 (COUNT47)
Lösung siehe Abb. 4.11

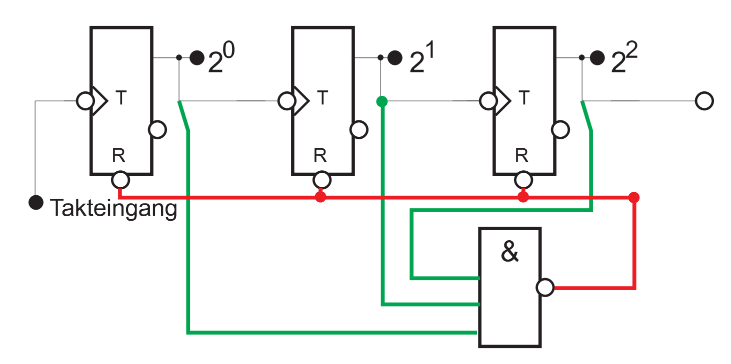


Abb. 4.9 Lösung zu Übung 4.11

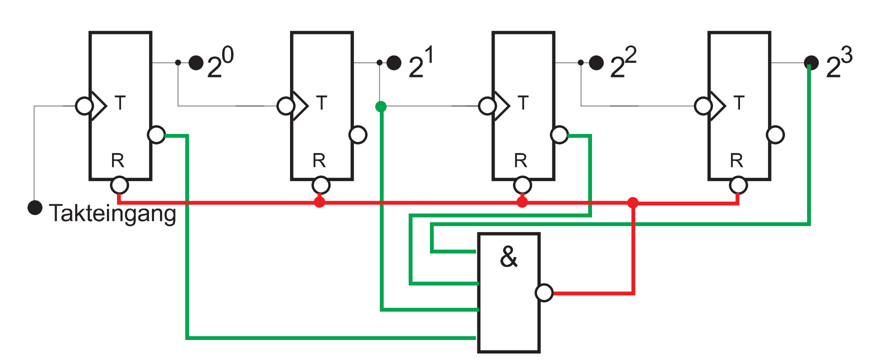


Abb. 4.10 Lösung zu Übung 4.12

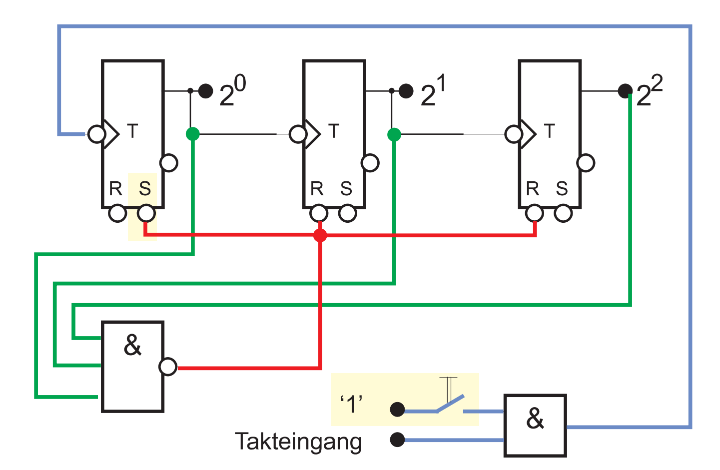


Abb. 4.11 Lösung zu Übung 4.13

Übung 4.14 (TANK42)
Im Buch fehlt bei der Schaltung das zweite T-FlipFlop oben.
Nach dem ersten Leeren wird das erste T-FlipFlop gesetzt. Der Rücksetzimpuls für das Start-FlipFlop wird erst nach dem zweiten Leeren wirksam, wenn das zweite T-FlipFlop gesetzt ist.
Wenn der Füllvorgang nicht nur zwei-, sondern dreimal hintereinander erfolgen soll müssen durch ein NAND-Glied *beide* T-FlipFlop-Ausgänge ausgewertet werden.

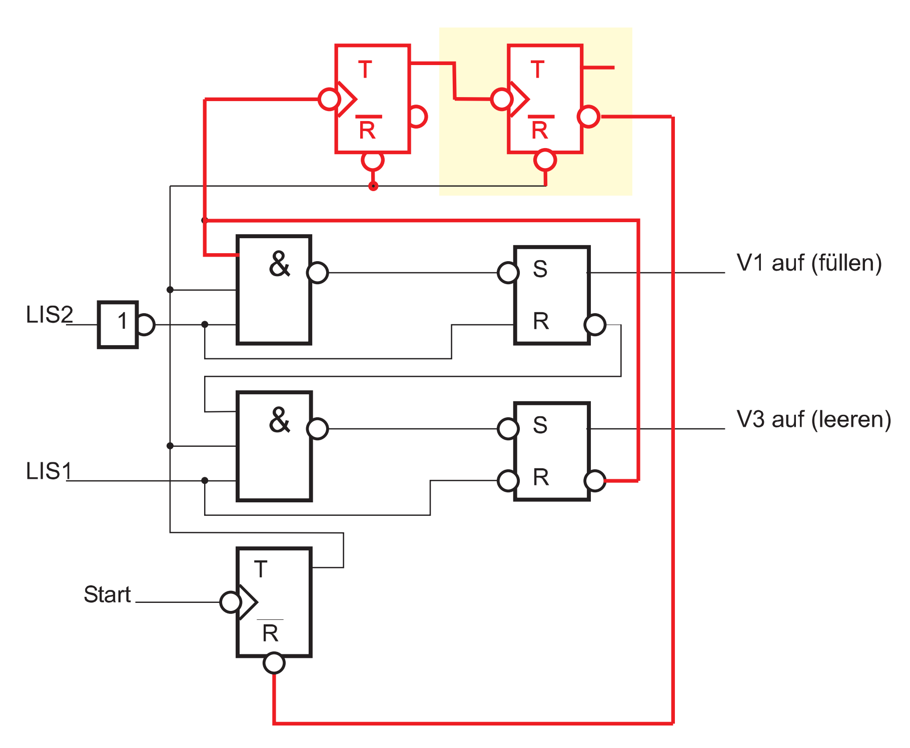


Abb. 4.12 Schaltung zu Übung 4.14

Übung 4.15 Für eine Zeit von 3 Minuten = 180 Sekunden werden bei einer Taktzeit von 0,5 Sekunden 360 Takte benötigt. Ein Zähler, der bis 360 zählen kann, benötigt 9 FlipFlop. $360_{Dez} = 101101000_{bin}$. Wenn die Taktzeit 0,1 Sekunde beträgt, muß der Zähler auf $1800_{Dez} = 11100001000_{bin}$ zählen. Es sind also 11 FlipFlop nötig.

Übung 4.16 (MIXER41)
Keine Musterlösung.

Teil II
SPS-Technik

Im zweiten Teil des Buches behandeln wir die SPS-Technik.

Kapitel 5
Schaltnetze mit SPS

Auf der Webseite[1] der Autoren zu diesem Buch steht das Programm PLC-lite zur Verfügung, mit dessen Hilfe Sie alle Übungen durchführen können.

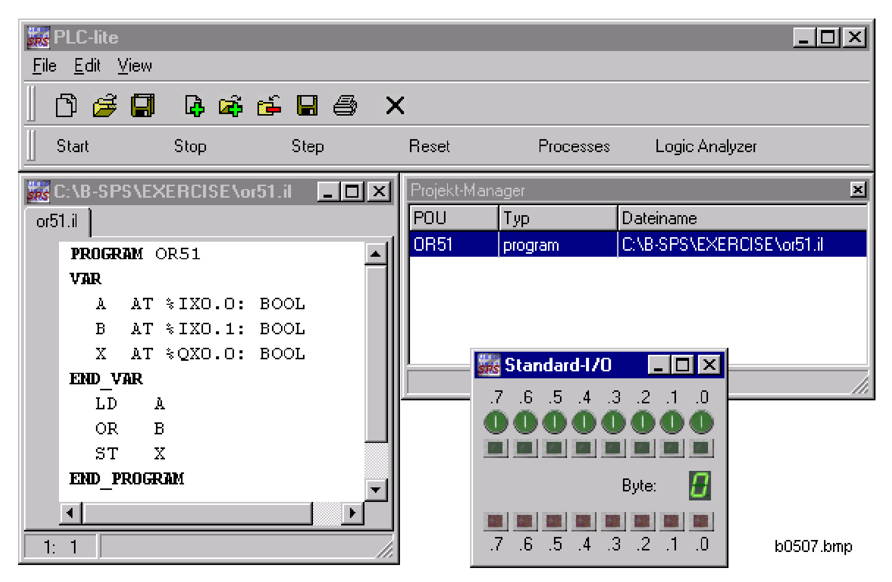


Abb. 5.1 PLC-lite: Hauptfenster, Editorfenster und Projektmanager

Nach dem Start des Programmes öffnen sich zunächst drei Fenster: das Hauptfenster mit den Bedienelementen zum Steuern des Projektes und der simulierten SPS, sowie das Editorfenster mit dem Editor, in dem Sie die Anweisungsliste eintippen können und das Fenster mit dem Projektmanager.

Klicken Sie im Hauptfenster auf den Button „Visualization“ (bzw. „Processes“ in älteren Versionen von PLC-lite) und wählen Sie zunächst „Standard-I/O“ aus. Über

[1] http://www.adamis.de/sps/

dieses Menü können Sie in späteren Übungen die zugehörigen Anzeigeelemente und Prozessmodelle öffnen.

Zum Einstellen der Eingangswerte für die SPS klicken Sie mit der Maus auf die grünen Taster. Die unteren, roten „Leuchtdioden" (LEDs) zeigen Ihnen den Ausgangswert. Eine „leuchtende" LED bedeutet eine logische ‚1'.

Die Eingänge wirken als Taster, wenn Sie mit der Maus klicken. Wenn Sie die Maus bei gedrückt gehaltener linker Maustaste von dem Taster wegziehen, bleibt er „eingerastet". Der Eingang bleibt dann auf Dauer eingeschaltet, bis Sie wieder auf dem Eingang klicken. Somit haben Sie einen „Schalter" am Eingang.

Übung 5.1 (OR51)

Verwenden in PLC-lite zur Visualisierung den Prozess „Standard I/O". Klicken Sie in der oberen Reihe die „Taster". Es leuchten die dem jeweiligen Eingang zugehörigen grünen LEDs auf, und in der unteren Reihe die dem Ausgang zugeordneten roten LEDs.

```
PROGRAM OR51
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
VAR
  a AT %IX0.0: BOOL
  b AT %IX0.1: BOOL
  x AT %QX0.0: BOOL
END_VAR
  LD a
  OR b
  ST x
END_PROGRAM
```

Übung 5.2 (AND51)

Beachten Sie bei der Simulation mit PLC-lite, dass Sie die Taster „festklemmen" können, indem Sie mit der Maus auf den Taster fahren, die linke Maustaste drücken, festhalten und dann herunter fahren. Der Taster bleibt dadurch gedrückt.

Eine nützliche Funktion ist der *Einzelschrittmodus*. Ein Klick auf ‚Step' führt in der Anweisungsliste den nächsten Befehl aus und wartet dann. Ein Klick auf ‚Cycle' arbeitet alle Anweisungen einmal durch und stoppt am Anfang der Anweisungsliste.

Diejenige Anweisung, welche beim nächsten Schritt ausgeführt werden wird, wird im Editor hervorgehoben dargestellt. Im sich gleichzeitig öffnenden Fenster ‚Watch expressions' werden Ihnen die aktuellen Werte der Variablen und des Aktuellen Ergebnis (`CR`) angezeigt.

```
PROGRAM AND51
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
VAR
```

```
  A  AT %IX0.0: BOOL
  B  AT %IX0.1: BOOL
  X  AT %QX0.0: BOOL
END_VAR
  LD   A
  AND  B
  ST   X
END_PROGRAM
```

Übung 5.3 (MIXER51)
Zum Füllen bzw. Leeren des Kessels von Hand, bitte die Taster neben V3 bzw. V4 drücken.

```
PROGRAM MIXER51
(*Processes:                *)
(*  Boiler                  *)
VAR
  MixerSwitch  AT %IX0.0: BOOL
  HeaterButton AT %IX0.7: BOOL
  Mixer AT %QX0.6: BOOL
  Heating AT %QX0.5: BOOL
  V4    AT %QX0.4: BOOL
END_VAR
  LD   MixerSwitch
  AND  HeaterButton
  STN  V4
  LD   MixerSwitch
  ST   Mixer
  LD   HeaterButton
  ST   Heating
END_PROGRAM
```

Übung 5.4 (NAND51)

```
PROGRAM NAND51
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
VAR
  A  AT %IX0.0: BOOL
  B  AT %IX0.1: BOOL
  X  AT %QX0.0: BOOL
  Y  AT %QX0.1: BOOL
END_VAR
  LD   A
```

```
    AND  B
    STN  X
    LDN  A
    ORN  B
    ST   Y
  END_PROGRAM
```

Übung 5.5 (NOR51)

```
  PROGRAM NOR51
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  VAR
    A  AT %IX0.0: BOOL
    B  AT %IX0.1: BOOL
    X  AT %QX0.0: BOOL
    Y  AT %QX0.1: BOOL
  END_VAR
    LD   A
    OR   B
    STN  X
    LDN  A
    ANDN B
    ST   Y
  END_PROGRAM
```

Übung 5.6 (BOILER51)

```
  PROGRAM Boiler51
  (*Processes:                *)
  (*  Boiler                  *)
  VAR
    Sw1  AT %IX0.0: BOOL
    TIC  AT %IX0.5: BOOL
    Heat AT %QX0.5: BOOL
  END_VAR
    LD   Sw1
    ANDN TIC
    ST   Heat
  END_PROGRAM
```

Übung 5.7 (BOILER52)

```
  PROGRAM Boiler52
  (*Processes:                *)
```

```
(*  Boiler                  *)
VAR
  Sw1  AT %IX0.0: BOOL
  TIC  AT %IX0.5: BOOL
  Heat AT %QX0.5: BOOL
  Start AT %IX0.7: BOOL
  Mixer AT %QX0.6: BOOL
END_VAR
  LD   Sw1
  ANDN TIC
  ST   Heat
  LD   Start
  ST   Mixer
END_PROGRAM
```

Übung 5.8 (BOILER53)
Zum Füllen bzw. Leeren des Kessels von Hand, bitte die Taster neben V3 bzw. V4 drücken.

```
PROGRAM Boiler53
(*Processes:                *)
(*  Boiler                  *)
VAR
  Sw1  AT %IX0.0: BOOL
  TIC  AT %IX0.5: BOOL
  Heat AT %QX0.5: BOOL
  Start AT %IX0.7: BOOL
  Mixer AT %QX0.6: BOOL
  LIS3  AT %IX0.3: BOOL
  Warn  AT %QX0.7: BOOL
END_VAR
  LD   Sw1
  ANDN TIC
  ST   Heat
  LD   Start
  ST   Mixer
  LDN  LIS3
  AND  Start
  ST   Warn
END_PROGRAM
```

Übung 5.9 (BOILER54)

```
PROGRAM Boiler54
(*Processes:                *)
```

```
(*  Boiler                  *)
VAR
  Sw1  AT %IX0.0: BOOL
  TIC  AT %IX0.5: BOOL
  Heat AT %QX0.5: BOOL
  Start AT %IX0.7: BOOL
  Mixer AT %QX0.6: BOOL
  LIS3  AT %IX0.3: BOOL
  Warn  AT %QX0.7: BOOL
  On_LED AT %QX0.0: BOOL
END_VAR
  LD   Sw1
  ANDN TIC
  ST   Heat
  LD   Start
  ST   Mixer
  LDN  LIS3
  AND  Start
  ST   Warn
  LD   Heat
  ST   On_LED
END_PROGRAM
```

Übung 5.10 (POWER51)

```
PROGRAM POWER51a
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
VAR
  a AT %IX0.0: bool
  b AT %IX0.1: bool
  c AT %IX0.2: bool
  x AT %QX0.0: bool
  y AT %QX0.1: bool
END_VAR
  LD  a
  OR  b
  ST  x
  LD  x
  AND c
  ST  y
END_PROGRAM
```

Weil das Aktuelle Ergebnis (`CR`) erhalten bleibt, ist die Zwischenspeicherung auf ’`x`’ nicht erforderlich. Das können sie mittels des Einzelschrittmodus in PLC-lite nachvollziehen: wiederholt den Button “Step“ klicken und im Fenster „Watch expressions“ die Wertezuweisungen verfolgen.

```
PROGRAM POWER51b
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
VAR
  a AT %IX0.0: bool
  b AT %IX0.1: bool
  c AT %IX0.2: bool
  y AT %QX0.1: bool
END_VAR
  LD  a
  OR  b
  AND c
  ST  y
END_PROGRAM
```

Übung 5.11 (POWER52)
Die Lösung entspricht genau „Power51b“!

```
Program POWER52
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
var
  a AT %IX0.0: bool
  b AT %IX0.1: bool
  c AT %IX0.2: bool
  y AT %QX0.0: bool
end_var
  LD  a
  OR  b
  AND c
  ST  y
end_program
```

Übung 5.12 (MEMO51)

```
Program Memo51
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
var
  a AT %IX0.0: bool
  b AT %IX0.1: bool
  c AT %IX0.2: bool
  mem : bool
  x AT %QX0.0: bool
  y AT %QX0.1: bool
```

```
  z AT %QX0.2: bool
end_var
(* Term 1 *)
  ld  a
  or  b
  st  mem
  ld  c
  and mem
  st  x
(* Term 2 *)
  ld  a
  or  b
  and c
  st  y
(* Term 3 *)
  ld   c
  and( a
  or   b
  )
  st   z
end_program
```

Übung 5.13 (EXOR51)

```
PROGRAM EXOR51
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
VAR
  A  AT %IX0.6: BOOL
  B  AT %IX0.7: BOOL
  X  AT %QX0.6: BOOL
END_VAR
  LD  A
  XOR B
  ST  X
END_PROGRAM
```

Kapitel 6
Schaltungen mit Signalspeichern

Übung 6.1 (POWER61)

```
Program POWER61
var
  a AT %IX0.0: bool
  b AT %IX0.1: bool
  x AT %QX0.0: bool
  y AT %QX0.1: bool
end_var
  LD   b
  OR   x
  ANDN a
  ST   x

  LD   y
  ANDN a
  OR   b
  ST   y
end_program
```

Übung 6.2 (RS61)

```
Program RS61
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Boiler                  *)
var
  a  AT %IX0.7: bool
  b  AT %IX0.6: bool
  x  AT %QX0.0: bool
end_var
```

```
    ld a
    s  x
    ld b
    r  x
  end_program
```

Übung 6.3 Keine Musterlösung.

Übung 6.4 Keine Musterlösung.

Übung 6.5 (RS62)

```
  Program RS62
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  var
    a  AT %IX0.7: bool
    b  AT %IX0.6: bool
    x  AT %QX0.0: bool
    y  AT %QX0.2: bool
    Memo1 : bool
    Memo2 : bool
  end_var
    ld a
    s  Memo1
    ld b
    r  Memo1
    ld Memo1
    st x

    ld b
    r  Memo2
    ld a
    s  Memo2
    ld Memo2
    st y
  end_program
```

Übung 6.6 (zu Power61)
Stromlaufplan A: vorrangig rücksetzend
Stromlaufplan B: vorrangig setzend

Übung 6.7 (TANK61)

```
Program Tank61
(*Processes:                *)
(*  Levelling               *)
VAR
  LIS1 AT %ix0.1: BOOL
  LIS2 AT %ix0.2: BOOL
  V1   AT %qx0.1: BOOL
END_VAR
  LDN   LIS1
  S     V1
  LD    LIS2
  R     V1
END_PROGRAM
```

Übung 6.8 (ALARM61)

```
Program Alarm61
(*Processes:                *)
(*  Boiler - press Heating to cause Fault  *)
(*OR                        *)
(*  Standard I/O  byte1     *)
(*  Panel                   *)
VAR
  Quit  AT %ix0.6: BOOL
  Fault AT %ix1.0: BOOL
  Horn  AT %qx0.7: BOOL
END_VAR
  LD   Fault
  S    Horn
  LD   Quit
  R    Horn
END_PROGRAM
```

Übung 6.9 (ALARM62)

```
Program Alarm62
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Standard-I/O  byte 1    *)
(*  Panel                   *)
VAR
  Tast0  AT %ix0.0: BOOL
  Tast1  AT %ix0.1: BOOL
  Tast2  AT %ix0.2: BOOL
  Quitt0 AT %ix1.0: BOOL
```

```
    Quitt1 AT %ix1.1: BOOL
    Quitt2 AT %ix1.2: BOOL
    Horn   AT %qx0.7: BOOL
    Alarm: BOOL
  END_VAR
    LD   Tast0
    OR   Tast1
    OR   Tast2
    S    Alarm

    LD   Quitt0
    OR   Quitt1
    OR   Quitt2
    R    Alarm

    LD   Alarm
    ST   Horn
  END_PROGRAM
```

Übung 6.10 (ALARM63)

```
  Program Alarm63
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  (*  Standard-I/O  byte 1    *)
  (*  Panel                   *)
  VAR
    Tast0  AT %ix0.0: BOOL
    Tast1  AT %ix0.1: BOOL
    Tast2  AT %ix0.2: BOOL
    Quitt0 AT %ix1.0: BOOL
    Quitt1 AT %ix1.1: BOOL
    Quitt2 AT %ix1.2: BOOL
    Lamp0  AT %qx1.0: BOOL
    Lamp1  AT %qx1.1: BOOL
    Lamp2  AT %qx1.2: BOOL
    Horn   AT %qx0.7: BOOL
    Alarm: BOOL
  END_VAR
    LD   Tast0
    OR   Tast1
    OR   Tast2
    S    Alarm

    LD   Quitt0
    OR   Quitt1
```

```
    OR   Quitt2
    R    Alarm
    R    Lamp0
    R    Lamp1
    R    Lamp2

    LD   Tast0
    S    Lamp0
    LD   Tast1
    S    Lamp1
    LD   Tast2
    S    Lamp2

    LD   Alarm
    ST   Horn
  END_PROGRAM
```

Übung 6.11 (ALARM64)

```
program ALARM64
(*Processes:                *)
(*  Boiler - press Heating to cause Fault  *)
(*OR                        *)
(*  Standard I/O  byte1     *)
(*  Panel                   *)
var
  y1: rs
  y2: rs
  Quitt AT %ix0.6: bool
  Fault AT %ix1.0: bool
  Lamp  AT %qx1.0: bool
  Horn  AT %qx0.7: bool
end_var

  ld  Quitt
  and Fault
  st  y1.s
  ldn Fault
  st  y1.r1

  ldn y1.q1
  and Fault
  st  y2.s
  ld  Quitt
  st  y2.r1
```

```
    cal y1
    cal y2

    ld  y1.q1
    or  y2.q1
    st  Lamp
    ld  y2.q1
    st  Horn

  end_program
```

Übung 6.12 (TANK62)

```
  Program Tank62
  (*Processes:               *)
  (*  Tanks (small)          *)
  VAR
    V1FF : RS
    Fill AT %ix0.7: BOOL
    LIS1 AT %ix0.1: BOOL
    LIS2 AT %ix0.2: BOOL
    V1   AT %qx0.1: BOOL
  END_VAR
    ld   Fill
    st   V1FF.S
    ld   LIS2
    st   V1FF.R1

    cal  V1FF

    ld   V1FF.Q1
    st   V1
  END_PROGRAM
```

Übung 6.13 (TANK63)

```
  Program Tank63
  (*Processes:               *)
  (*  Tanks (small)          *)
  VAR
    V1FF : RS
    V3FF : RS
    Fill  AT %ix0.7: BOOL
    Empty AT %ix0.6: BOOL
```

```
    LIS1  AT %ix0.1: BOOL
    LIS2  AT %ix0.2: BOOL
    V1    AT %qx0.1: BOOL
    V3    AT %qx0.3: BOOL
  END_VAR
    ld   Fill
    st   V1FF.S
    ld   LIS2
    st   V1FF.R1

    ld Empty
    st   V3FF.S
    ldn  LIS1
    st   V3FF.R1

    cal  V1FF
    cal  V3FF

    ld   V1FF.Q1
    st   V1

    ld   V3FF.Q1
    st   V3
  END_PROGRAM
```

Übung 6.14 (TANK64)

```
  Program Tank64
  (*Processes:                *)
  (*  Tanks (small)           *)
  VAR
    V1FF : RS
    V3FF : RS
    Fill  AT %ix0.7: BOOL
    Empty AT %ix0.6: BOOL
    LIS1  AT %ix0.1: BOOL
    LIS2  AT %ix0.2: BOOL
    V1    AT %qx0.1: BOOL
    V3    AT %qx0.3: BOOL
  END_VAR
    ld   Fill
    andn V3FF.Q1
    st   V1FF.S
    ld   LIS2
    st   V1FF.R1
```

```
    ld   Empty
    andn V1FF.Q1
    st   V3FF.S
    ldn  LIS1
    st   V3FF.R1

    cal  V1FF
    cal  V3FF

    ld   V1FF.Q1
    st   V1

    ld   V3FF.Q1
    st   V3
  END_PROGRAM
```

Übung 6.15 (TANK65)

```
  Program Tank65
  (*Processes:                 *)
  (*  Tanks (small)            *)
  VAR
    V1FF : RS
    V3FF : RS
    Fill  AT %ix0.7: BOOL
    LIS1  AT %ix0.1: BOOL
    LIS2  AT %ix0.2: BOOL
    V1    AT %qx0.1: BOOL
    V3    AT %qx0.3: BOOL
  END_VAR
    ld   Fill
    andn V3FF.Q1
    st   V1FF.S
    ld   LIS2
    st   V1FF.R1

    ld   LIS2
    andn V1FF.Q1
    st   V3FF.S
    ldn  LIS1
    st   V3FF.R1

    cal  V1FF
    cal  V3FF

    ld   V1FF.Q1
```

```
    st   V1

    ld   V3FF.Q1
    st   V3
  END_PROGRAM
```

Übung 6.16 (SWITCH61) Keine Musterlösung.

Kapitel 7
Zeitfunktionen mit SPS

Übung 7.1 (MIXER71)

```
Program Mixer71
(*Processes:                *)
(*  Boiler                  *)
var
  Start AT %ix0.7: BOOL
  Mixer AT %qx0.6: BOOL
  Impuls: TP
end_var
  ld Start
  st Impuls.IN
  ld t#3000ms
  st Impuls.PT

  cal Impuls

  ld Impuls.Q
  st Mixer
end_program
```

Übung 7.2 (MIXER72)

```
Program Mixer72
(*Processes:                *)
(*  Boiler                  *)
var
  Start AT %ix0.7: BOOL
  Mixer AT %qx0.6: BOOL
  Impuls: TP
  Lamp AT %q0.0: BOOL
end_var
```

```
    ld Start
    st Impuls.IN
    ld t#3000ms
    st Impuls.PT

    cal Impuls

    ld  Impuls.Q
    st  Mixer
    st  Lamp
  end_program
```

Übung 7.3 (MIXER73)

```
  Program Mixer73
  (*Processes:                *)
  (*  Boiler                  *)
  var
    Start AT %ix0.7: BOOL
    Stop  AT %ix0.6: BOOL
    Mixer AT %qx0.6: BOOL
    Lamp  AT %qx0.0: BOOL
    Impuls: TP
    Run: BOOL
  end_var
    ld Start
    s  Run
    ld Stop
    r  Run

    ld Start
    st Impuls.IN
    ld t#3000ms
    st Impuls.PT

    cal Impuls

    ld  Impuls.Q
    or  Start
    and Run
    st  Mixer

    ld  Run
    st  Lamp
  end_program
```

Übung 7.4 (TANK71)

```
Program Tank71
(*Processes:                *)
(*  Tanks (small)           *)
VAR
  V1FF : RS
  V3FF : RS
  Fill  AT %ix0.7: BOOL
  Empty AT %ix0.6: BOOL
  LIS1  AT %ix0.1: BOOL
  LIS2  AT %ix0.2: BOOL
  V1    AT %qx0.1: BOOL
  V3    AT %qx0.3: BOOL
  Horn  AT %qx0.7: BOOL
  Pause : TP
END_VAR
  ld   Fill
  andn V3FF.Q1
  st   V1FF.S
  ld   LIS2
  st   V1FF.R1
  st   Pause.IN

  ld   t#5000ms
  st   Pause.PT
  cal  Pause

  ld   Pause.Q
  st   Horn

  ld   Empty
  andn Pause.Q
  andn V1FF.Q1
  st   V3FF.S

  ldn  LIS1
  st   V3FF.R1

  cal  V1FF
  cal  V3FF

  ld   V1FF.Q1
  st   V1
  ld   V3FF.Q1
  st   V3
```

```
END_PROGRAM
```

Übung 7.5 (TANK72)

```
Program Tank72
(*Processes:                   *)
(*  Tanks (small)              *)
VAR
  V1FF : RS
  V3FF : RS
  Fill  AT %ix0.7: BOOL
  Empty AT %ix0.6: BOOL
  LIS1  AT %ix0.1: BOOL
  LIS2  AT %ix0.2: BOOL
  V1    AT %qx0.1: BOOL
  V3    AT %qx0.3: BOOL
  Horn  AT %qx0.7: BOOL
  Pause : TP
  TimeAfter: TP
  isEmpty: BOOL
  isFull:  BOOL
END_VAR
  ldn  LIS1
  st   isEmpty

  ld   LIS2
  st   isFull

  ld   Fill
  andn V3FF.Q1
  st   V1FF.S
  ld   isFull
  st   V1FF.R1
  st   Pause.IN

  ld   t#3s
  st   Pause.PT
  cal  Pause

  ld   Pause.Q
  st   Horn

  ld   Empty
  andn Pause.Q
  andn V1FF.Q1
  st   V3FF.S
```

```
    ld   isEmpty
    st   TimeAfter.IN

    ld   t#3s
    st   TimeAfter.PT
    cal  TimeAfter

    ld   isEmpty
    andn TimeAfter.Q
    st   V3FF.R1

    cal  V1FF
    cal  V3FF

    ld   V1FF.Q1
    st   V1
    ld   V3FF.Q1
    st   V3
  END_PROGRAM
```

Übung 7.6 (FLASH71)

```
  program GeneratorERROR_Flash71
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  var
    Impuls1:        TP
    Impuls2:        TP
    Lamp AT %QX0.0: bool
  end_var
    ld  t#1000ms    (* Zeiten einstellen        *)
    st  Impuls1.pt
    st  Impuls2.pt
    ldn Impuls2.q   (* läuft Timer2?            *)
    st  Impuls1.in  (* wenn nein Timer1 starten *)
    ldn Impuls1.q   (* läuft Timer1?            *)
    st  Impuls2.in  (* wenn nein Timer2 starten *)
    cal Impuls1     (* Timer aufrufen           *)
    cal Impuls2
    ld  Impuls1.q
    st  Lamp
  end_program
```

Übung 7.7 (FLASH72)

```
program Flash72
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
var
  Impuls1:        TP
  Impuls2:        TP
  Lamp AT %QX0.0: bool
end_var
  ld  t#1000ms    (* Zeiten einstellen        *)
  st  Impuls1.pt
  st  Impuls2.pt
  ldn Impuls2.q   (* läuft Timer2?            *)
  st  Impuls1.in  (* wenn nein Timer1 starten *)
  cal Impuls1     (* Timer1 aufrufen          *)

  ldn Impuls1.q   (* läuft Timer1?            *)
  st  Impuls2.in  (* wenn nein Timer2 starten *)
  cal Impuls2     (* Timer2 aufrufen          *)

  ld  Impuls1.q
  st  Lamp
end_program
```

Übung 7.8 (ALARM71)

```
program ALARM71
(*Processes:                *)
(*  Boiler - press Heating to cause Fault  *)
(*OR                        *)
(*  Standard I/O  byte1     *)
(*  Panel                   *)
var
  y1: rs
  y2: rs
  Quitt AT %ix0.6: bool
  Fault AT %ix1.0: bool
  Lamp  AT %qx1.0: bool
  Horn  AT %qx0.7: bool
  Impuls1: TP
  Impuls2: TP
  y3: bool
end_var

  ld  Quitt
  and Fault
```

```
    st  y1.s
    ldn Fault
    st  y1.r1

    ldn y1.q1
    and Fault
    st  y2.s
    ld  Quitt
    st  y2.r1

    cal y1
    cal y2

    ld t#1250ms
    st Impuls1.pt
    st Impuls2.pt

    ldn Impuls2.q
    st  Impuls1.in
    cal Impuls1
    ldn Impuls1.q
    st  Impuls2.in
    cal Impuls2

    ld  y1.q1
    or  y3
    st  Lamp

    ld  y2.q1
    and Impuls1.Q
    st  y3

    ld  y3
    st  Horn

  end_program
```

Übung 7.9 (ALARM72)

```
  program ALARM72
  (*Processes:                *)
  (*  Boiler - press Heating to cause Fault  *)
  (*OR                        *)
  (*  Standard I/O  byte1     *)
  (*  Panel                   *)
  var
```

```
    y1: rs
    y2: rs
    Quitt AT %ix0.6: bool
    Fault AT %ix1.0: bool
    Lamp  AT %qx1.0: bool
    Horn  AT %qx0.7: bool
    Impuls1: TP
    Impuls2: TP
    Impuls3: TP
  end_var

    ld  Quitt
    and Fault
    st  y1.s
    ldn Fault
    st  y1.r1

    ldn y1.q1
    and Fault
    st  y2.s
    ld  Quitt
    st  y2.r1

    cal y1
    cal y2

    ld t#1250ms
    st Impuls1.pt
    st Impuls2.pt
    ld  t#5s
    st  Impuls3.pt

    ldn Impuls2.q
    st  Impuls1.in
    cal Impuls1
    ldn Impuls1.q
    st  Impuls2.in
    cal Impuls2

    ld  Quitt
    st  Impuls3.IN
    cal Impuls3

    ld  y2.q1
    andn Impuls1.Q
```

```
  or  Impuls3.Q
  st  Lamp

  ld  y2.q1
  and Impuls1.Q
  st  Horn

end_program
```

Übung 7.10 (FLASH73)

```
program Flash73
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
var
  Impuls1:         TP
  Impuls2:         TP
  Impuls3:         TP
  Lamp1  AT %QX0.0: BOOL
  Lamp2  AT %QX0.1: BOOL
  Lamp3  AT %QX0.2: BOOL
end_var
  ld t#1s
  st Impuls1.pt
  st Impuls2.pt
  st Impuls3.pt

  ldn Impuls3.q
  andn Impuls2.q
  st  Impuls1.in
  cal Impuls1

  ldn Impuls1.q
  andn Impuls3.q
  st  Impuls2.in
  cal Impuls2

  ldn Impuls2.q
  andn Impuls1.q
  st  Impuls3.in
  cal Impuls3

  ld  Impuls1.q
  st  Lamp1
  ld  Impuls2.q
```

```
    st  Lamp2
    ld  Impuls3.q
    st  Lamp3
  end_program
```

Übung 7.11 (FLASH75)

```
  Program Einschaltverz
  (*Processes:                *)
  (*  Boiler                  *)
  var
    Pulse: TON
    Start AT %IX0.0: bool
    Lamp  AT %QX0.0: bool
  end_var
    ld   Start
    st   Pulse.IN
    ld   t#1000ms
    st   Pulse.PT
    cal  Pulse
    ld   Pulse.Q
    st   Lamp
  end_program
```

Übung 7.12 (FLASH76)

```
  PROGRAM IMPULSE
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  VAR
    PULSE: TON
    LAMP AT %QX0.0: BOOL
  END_VAR
    LDN PULSE.Q
    ST  PULSE.IN
    LD  T#500MS
    ST  PULSE.PT
    CAL PULSE
    LD  PULSE.Q
    ST  LAMP
  END_PROGRAM
```

Übung 7.12 (FLASH76long)

```
Program ImpulseLang
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
var
  LED:   TP     (* Impulsverlängerer           *)
  Pulse: TON
  Lamp AT %QX0.0: bool
end_var
  ldn Pulse.Q   (* Ausgang negiert laden und   *)
  st  Pulse.IN  (* auf den Eingang rückkoppeln *)
  ld  t#500ms
  st  Pulse.PT
  cal Pulse     (* Timer starten               *)
  ld  t#150ms   (* Pulsdauer verlängern        *)
  st  LED.PT
  ld  Pulse.Q
  st  LED.IN
  cal LED
  ld  LED.Q     (* Ausgang des LED-Timers      *)
  st  Lamp
end_program
```

Kapitel 8
Zähler mit SPS

Übung 8.1 (INOUT81)

```
PROGRAM INTEGER
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Hex-Output    byte 0    *)
VAR
  VALUE AT %IB0: sint
  OUT   AT %QB0: sint
END_VAR
  LD VALUE
  ST OUT
END_PROGRAM
```

Übung 8.2 (INOUT82)

```
PROGRAM INTEGER
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Standard-I/O  byte 1    *)
(*  Hex-Input     byte 0    *)
(*  Hex-Input     byte 1    *)
(*  Hex-Output    byte 0    *)
(*  Hex-Output    byte 1    *)
(*  dword: also bytes 2 & 3 *)
VAR
  VALUE AT %IB0: sint
  OUT   AT %QB0: sint
END_VAR
  LD VALUE
  ST OUT
END_PROGRAM
```

Übung 8.3 (INOUT83)

```
PROGRAM INTEGER
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Standard-I/O  byte 1    *)
(*  Hex-Input     byte 0    *)
(*  Hex-Input     byte 1    *)
(*  Hex-Output    byte 0    *)
(*  Hex-Output    byte 1    *)
(*  dword: also bytes 2 & 3 *)
VAR
  VALUE AT %IB0: usint
  OUT   AT %QB0: sint
END_VAR
  LD VALUE
  usint_to_sint
  ST OUT
END_PROGRAM
```

Übung 8.4 (COUNT81)

```
PROGRAM CountUP
(*Processes:                *)
(*  Counter CTU             *)
(*  Hex-Input     byte 1    *)
(*  Hex-Output    byte 1    *)
VAR
  Counter :   CTU
  Clock       AT %IX0.3: Bool
  Reset       AT %IX0.0: Bool
  ComparePV   AT %QX0.3: Bool
  PresetValue AT %IB1: SINT
  CountValue  AT %QB1: SINT
END_VAR
  LD  Clock
  ST  Counter.CU
  LD  PresetValue
  SINT_TO_INT    (* Typumwandlung! *)
  ST  Counter.PV
  LD  Reset
  ST  Counter.R
  CAL Counter    (* Zähler aufrufen *)
  LD  Counter.Q
  ST  ComparePV
```

```
    LD  Counter.CV
    INT_TO_SINT    (* Typumwandlung! *)
    ST  CountValue
  END_PROGRAM
```

Übung 8.5 (COUNT82)

```
  PROGRAM CountDown
  (*Processes:                *)
  (*  Counter CTD             *)
  (*  Hex-Input     byte 1    *)
  (*  Hex-Output    byte 1    *)
  VAR
    Counter : CTD
    Clock AT %IX0.2: Bool
    Load  AT %IX0.1: Bool
    ComparePV AT %QX0.2: Bool
    PresetValue AT %IB1: SINT
    CountValue  AT %QB1: SINT
  END_VAR
    LD  Clock
    ST  Counter.CD
    LD  PresetValue
    sint_to_int
    ST  Counter.PV
    LD  Load
    ST  Counter.LD

    CAL Counter

    LD  Counter.Q
    ST  ComparePV
    LD  Counter.CV
    int_to_sint
    ST  CountValue
  END_PROGRAM
```

Übung 8.6 (COUNT83)

```
  PROGRAM CountUPDN
  (*Processes:                *)
  (*  Counter CTUD            *)
  (*  Hex-Input     byte 1    *)
```

```
(*  Hex-Output    byte 1    *)
VAR
  Counter : CTUD
  ClockDN     AT %IX0.2: Bool
  ClockUP     AT %IX0.3: Bool
  Reset       AT %IX0.0: Bool
  SetLoad     AT %IX0.1: Bool
  CompareZero AT %QX0.2: Bool
  ComparePV   AT %QX0.3: Bool
  PresetValue AT %IB1:   SINT
  CountValue  AT %QB1:   SINT
END_VAR

  LD  PresetValue
  SINT_TO_INT    (* Typumwandlung! *)
  ST  Counter.PV
  LD  ClockDN
  ST  Counter.CD
  LD  ClockUP
  ST  Counter.CU
  LD  Reset
  ST  Counter.R
  LD  SetLoad
  ST  Counter.LD

  CAL Counter

  LD  Counter.QU
  ST  ComparePV
  LD  Counter.QD
  ST  CompareZero
  LD  Counter.CV
  INT_TO_SINT    (* Typumwandlung! *)
  ST  CountValue
END_PROGRAM
```

Übung 8.7 (COUNT84)

```
PROGRAM Count84
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Hex-Output    byte 1    *)
VAR
  Counter : CTUD
  ClientOUT AT %IX0.0: Bool
```

```
    ClientIN  AT %IX0.1: Bool
    NoClient     AT %QX0.0: Bool
    Client1to5   AT %QX0.1: Bool
    MoreThanFive AT %QX0.2: Bool
    CountValue   AT %QB1:   SINT
  END_VAR
    LD  5
    ST  Counter.PV
    LD  ClientOUT
    ST  Counter.CD
    LD  ClientIN
    ST  Counter.CU

    CAL Counter

    LD  Counter.QU
    ST  MoreThanFive
    LD  Counter.QD
    ST  NoClient

    LD  MoreThanFive
    XOR NoClient
    STN Client1to5
    LD  Counter.CV
    INT_TO_SINT    (* Typumwandlung! *)
    ST  CountValue
  END_PROGRAM
```

Übung 8.8 (DEKADE81)

```
  PROGRAM Dekade81
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  (*  Standard-I/O  byte 1    *)
  (*  Standard-I/O  byte 2    *)
  (*  Standard-I/O  byte 3    *)
  (*  Hex-Output    byte 1    *)
  (*  Hex-Output    byte 2    *)
  (*  Hex-Output    byte 3    *)
  VAR
    One : CTU
    Ten : CTU
    Hun : CTU
    Clock AT %IX0.3: Bool
    Reset AT %IX0.0: Bool
```

```
    ByteOne AT %QB1: SINT
    ByteTen AT %QB2: SINT
    ByteHun AT %QB3: SINT

   END_VAR
    LD  10
    ST  One.PV
    ST  Ten.PV
    ST  Hun.PV
    LD  Clock
    ST  One.CU

    CAL One
    LD  One.CV
    int_to_sint
    ST  ByteOne
    LD  One.Q
    ST  Ten.CU
    OR  Reset
    ST  One.R

    CAL Ten
    LD  Ten.CV
    int_to_sint
    ST  ByteTen
    LD  Ten.Q
    ST  Hun.CU
    OR  Reset
    ST  Ten.R

    CAL Hun
    LD  Hun.CV
    int_to_sint
    ST  ByteHun
    LD  Hun.Q
    OR  Reset
    ST  Hun.R

  END_PROGRAM
```

Übung 8.9 (MIXER81)

```
  Program Mixer81
  (*Processes:                *)
  (*  Chemical Process(small) *)
```

```
(*  - OR -  Tanks (small)   *)
VAR
  V1FF : RS
  V3FF : RS
  Start AT %ix0.7: BOOL
  Stop  AT %ix0.6: BOOL
  LIS1  AT %ix0.1: BOOL
  LIS2  AT %ix0.2: BOOL
  V1    AT %qx0.1: BOOL
  V3    AT %qx0.3: BOOL
  Lamp  AT %qx0.0: BOOL
  Value AT %qb3: SINT
  Run : BOOL
  Fill: BOOL
  Counter: CTD
END_VAR
  ld   3
  st   Counter.PV

  ld   Start
  and  Counter.Q
  S    Run
  S    Fill
  st   Counter.LD

  cal  Counter

  ld   Counter.Q
  R    Fill
  ld   Counter.CV
  int_to_sint
  st   Value

  ld   Fill
  andn V3FF.Q1
  st   V1FF.S
  ld   LIS2
  st   V1FF.R1

  ld   LIS2
  andn V1FF.Q1
  st   V3FF.S
  ldn  LIS1
  st   V3FF.R1
  st   Counter.CD
```

```
    cal  V1FF
    cal  V3FF

    ld   V1FF.Q1
    st   V1

    ld   V3FF.Q1
    st   V3

    ld   Run
    st   Lamp
  END_PROGRAM
```

Übung 8.10 (GEN81)

```
  Program ImpulseGEN81
  (*Processes:                *)
  (*  Hex-Output    byte 3    *)
  var
    Display AT %qb3: SINT
    Zaehler: CTU
    Pulse: TON
  end_var
  (* Zaehler *)
    ld Pulse.Q
    st Zaehler.CU

  (* Impulsgenerator *)
    ldn Pulse.Q
    st Pulse.IN
    ld t#500ms
    st Pulse.PT

    cal Zaehler
    cal Pulse

    ld Zaehler.CV
    int_to_sint
    st Display
  end_program
```

Übung 8.11 (GEN82)

```
  Program ImpulseGEN82
```

```
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Hex-Output    byte 3    *)
var
  Display AT %qb3: SINT
  Counter: CTU
  Pulse: TON
  Stopp AT %i0.7: BOOL
end_var
(* Zaehler *)
  ld Pulse.Q
  st Counter.CU

(* Impulsgenerator *)
  ldn Pulse.Q
  andn Stopp
  st Pulse.IN
  ld t#100ms
  st Pulse.PT

  cal Counter
  cal Pulse

  ld Counter.CV
  int_to_sint
  st Display
end_program
```

Übung 8.12 (GEN83)

```
Program ImpulseGEN83
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Hex-Output    byte 3    *)
(*  -  OR  -                *)
(*  Chemical Process (small)*)
var
  Display AT %qb3: SINT
  Counter: CTUD
  Pulse: TON
  Pulse2:TON
  runUP AT %i0.7: BOOL
  runDN AT %i0.6: BOOL
end_var
(* Zaehler *)
  ld Pulse.Q
```

```
  st Counter.CU
  ld Pulse2.Q
  st Counter.CD

(* Impulsgenerator *)
  ldn Pulse.Q
  and runUP
  st Pulse.IN
  ld t#100ms
  st Pulse.PT

(* Impulsgenerator2 *)
  ldn Pulse2.Q
  and runDN
  st Pulse2.IN
  ld t#100ms
  st Pulse2.PT

  cal Counter
  cal Pulse
  cal Pulse2

  ld Counter.CV
  int_to_sint
  st Display
end_program


Program ImpulseGEN83b
(*Processes:                 *)
(*  Standard-I/O  byte 0     *)
(*  Hex-Input     byte 3     *)
(*  Hex-Output    byte 3     *)
var
  Display AT %qb3: SINT
  MaxVal  AT %ib3: SINT
  Counter: CTUD
  Pulse: TON
  Pulse2:TON
  runUP AT %i0.7: BOOL
  runDN AT %i0.6: BOOL
end_var
(* Zaehler *)
  ld Pulse.Q
  st Counter.CU
  ld Pulse2.Q
```

```
    st Counter.CD
    ld MaxVal
    sint_to_int
    st Counter.PV

  (* Impulsgenerator *)
    ldn Pulse.Q
    and runUP
    andn Counter.QU
    st Pulse.IN
    ld t#100ms
    st Pulse.PT

  (* Impulsgenerator2 *)
    ldn Pulse2.Q
    and runDN
    andn Counter.QD
    st Pulse2.IN
    ld t#100ms
    st Pulse2.PT

    cal Counter
    cal Pulse
    cal Pulse2

    ld Counter.CV
    int_to_sint
    st Display
  end_program
```

Übung 8.13 (TIME81)

```
  Program Time81
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  (*  Hex-Output    byte 3    *)
  var
    L1 AT %ix0.1: BOOL
    L2 AT %ix0.2: BOOL
    Stop  AT %ix0.6: BOOL
    SW1   AT %ix0.0: BOOL
    Display AT %qb3: SINT
    Counter: CTU
    Pulse: TON
    count: bool
```

```
end_var
(* Zaehler *)
  ld Pulse.Q
  st Counter.CU

(* Starten *)
  ld L1
  s count
(* Stoppen *)
  ld L2
  r count
(* Zaehler Reset *)
  ld Stop
  or SW1
  st Counter.R

(* Impulsgenerator *)
  ldn Pulse.Q
  and count   (* Lauf *)
  st Pulse.IN
  ld t#200ms
  st Pulse.PT

  cal Counter
  cal Pulse

  ld Counter.CV
  int_to_sint
  st Display
end_program
```

Übung 8.14 (TIME82)

```
Program FillTimeCountTime82
(*Processes:                *)
(*  Chemical Process (small)*)
var
  LIS1 AT %ix0.1: BOOL
  LIS2 AT %ix0.2: BOOL
  Start AT %ix0.7: BOOL
  Stop  AT %ix0.6: BOOL
  SW1   AT %ix0.0: BOOL
  Display AT %qb3: SINT
  V1 AT %qx0.1: BOOL
  V3 AT %qx0.3: BOOL
  Counter: CTU
```

```
    Pulse: TON
    count: bool
    run: bool
  end_var
  (* Starten *)
    ld Start
    s run
    r count
  (* Stoppen *)
    ld Stop
    r run
  (* Fuellen *)
    ld run
    st V1
  (* Leeren *)
    ldn run
    st V3

  (* Zaehler *)
    ld Pulse.Q
    st Counter.CU

  (* Starten *)
    ld LIS1
    s count
  (* Stoppen *)
    ld LIS2
    r count
  (* Zaehler Reset *)
    ld Stop
    or SW1
    st Counter.R

  (* Impulsgenerator *)
    ldn Pulse.Q
    and run
    and count   (* Lauf *)
    st Pulse.IN
    ld t#10ms
    st Pulse.PT

    cal Counter
    cal Pulse

    ld Counter.CV
```

```
    int_to_sint
    st Display
  end_program
```

Übung 8.15 (TANK81)

```
  Program Tank81
  (*Processes:                *)
  (*  Tanks (small)           *)
  var
    SW1   AT %ix0.0: BOOL
    Start AT %ix0.7: BOOL
    Stop  AT %ix0.6: BOOL
    LIS1  AT %ix0.1: BOOL
    LIS2  AT %ix0.2: BOOL
    V1    AT %qx0.1: BOOL
    V3    AT %qx0.3: BOOL
    Display AT %qb3: SINT

    Zaehler: CTU
    Pulse: TON
    count: bool
    run: bool
  end_var
  (* Zaehler *)
    ld Pulse.Q
    st Zaehler.CU

  (* Starten *)
    ld LIS1
    and run
    s count
    ld Start
    s run
  (* Stoppen *)
    ld LIS2
    r count
    ld Stop
    r run
  (* Zaehler Reset *)
    ld SW1
    st Zaehler.R
    r count

  (* Fuellen *)
```

```
    ld run
    st V1
  (* Leeren *)
    ldn run
    st V3


  (* Impulsgenerator *)
    ldn Pulse.Q
    and count   (* Lauf *)
    st Pulse.IN
    ld 30
    st Pulse.PT

    cal Zaehler
    cal Pulse

    ld Zaehler.CV
    int_to_sint
    st Display
  end_program
```

Übung 8.16 (REAKT81)

```
  Program Reaktion
  (* Nach Druck auf Starttaster *)
  (* leuchtet nach Vorlaufzeit  *)
  (* Led2 an qx0.6 auf.         *)
  (* Gleichzeitig startet die Zeit*)
  (* Druck auf Stoptaster stoppt*)
  (* die Reaktionszeit          *)
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  (*  Hex-Output    byte 3    *)
  var
    Start AT %ix0.7: BOOL
    Stop  AT %ix0.6: BOOL
    SW1   AT %ix0.0: BOOL
    Display AT %qb3: SINT
    Led1 AT %qx0.7: BOOL
    Led2 AT %qx0.6: BOOL
    Counter: CTU
    Pulse: TON
    count: bool
    run: bool
```

```
    Vorlauf: TP
  end_var
  (* Zaehler *)
    ld Pulse.Q
    st Counter.CU
  (* Starten *)
    ldn Vorlauf.Q
    and run
    s count
    ld Start
    s run
  (* Stoppen *)
    ld Stop
    r count
    r run
  (* Z?hler Reset *)
    ld Start
    st Counter.R
  (* Vorlaufzeit *)
    ld run
    st Vorlauf.IN
    ldn Vorlauf.Q
    stn Led1
    and run
    st Led2
    ld 3000
    st Vorlauf.pt
  (* Impulsgenerator *)
    ldn Pulse.Q
    and count   (* Lauf *)
    st Pulse.IN
    ld t#1ms
    st Pulse.PT

    cal Counter
    cal Pulse
    cal Vorlauf

    ld Counter.CV
    int_to_sint
    st Display

  end_program
```

Kapitel 9
Funktionsbausteine

Übung 9.1

```
program Flash72
var
  Impuls1:        TP
  Impuls2:        TP
  Lamp AT %QX0.0: bool
end_var
  ld  t#1000ms    (* Zeiten einstellen        *)
  st  Impuls1.pt
  st  Impuls2.pt

  ldn Impuls2.q   (* läuft Timer2?            *)
  st  Impuls1.in  (* wenn nein Timer1 starten *)
  cal Impuls1     (* Timer1 aufrufen          *)

  ldn Impuls1.q   (* läuft Timer1?            *)
  st  Impuls2.in  (* wenn nein Timer2 starten *)
  cal Impuls2     (* Timer2 aufrufen          *)

  ld  Impuls1.q
  st  Lamp
end_program
```

```
FUNCTION_BLOCK Gen05
VAR
  Impuls1 : TP
  Impuls2 : TP
END_VAR
VAR_OUTPUT
  Y : BOOL
END_VAR
```

```
    LD   t#500ms
    ST   Impuls1.PT
    ST   Impuls2.PT

    LDN  Impuls2.Q
    ST   Impuls1.IN
    CAL  Impuls1

    LDN  Impuls1.Q
    ST   Impuls2.IN
    CAL  Impuls2

    LD   Impuls1.Q
    ST   Y
    RET
  END_FUNCTION_BLOCK
```

Übung 9.2 (FLASH92)

```
program FlashLightFlash92
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
var
  Flash: Gen05
  Lamp AT %qx0.0: Bool
end_var
  CAL Flash
  LD  Flash.Y
  ST  Lamp
end_program
```

Übung 9.3 (GEN91)

```
Program ImpulseGen91
(*Processes:                *)
(*  Hex-Output  byte 3      *)
var
  Display AT %qb3: SINT
  Counter: CTU
  Pulse: Sec1
end_var
(* Counter *)
  ld Pulse.Y
  st Counter.CU
```

```
(* Impulsgenerator *)
  cal Pulse
  cal Counter

  ld Counter.CV
  int_to_sint
  st Display
end_program

---

Function_Block Sec1
var
  Pulse: TON
end_var
var_output
  Y: Bool
end_var
(* Impulsgenerator *)
  ldn Pulse.Q
  st  Pulse.IN
  ld  t#1s
  st  Pulse.PT
  cal Pulse
  ld  Pulse.Q
  st  Y
end_function_block
```

Übung 9.4 (ALARM91)

```
Program Alarm91
(*Processes:                *)
(*  Boiler - press Heating to cause Fault  *)
(*OR                        *)
(*  Standard I/O  byte1     *)
(*  Panel                   *)
VAR
  Quit  AT %ix0.6: BOOL
  Fault AT %ix1.0: BOOL
  Horn  AT %qx0.7: BOOL
  Blink: Gen05
  Lamp  AT %qx1.0: BOOL
END_VAR
  LD   Fault
  S    Horn
```

```
    LD   Quit
    R    Horn
    CAL  Blink
    LD   Horn
    AND  Blink.Y
    ST   Lamp
  END_PROGRAM
```

Übung 9.5 Keine Musterlösung.

Übung 9.6 (FFFB92)

```
  (* FUNCTION BLOCK *)
  FUNCTION_BLOCK FFSR
  VAR_INPUT
    Set  : BOOL
    Rset : BOOL
  END_VAR
  VAR_OUTPUT
    Y : BOOL
  END_VAR
    LD    Set
    ORN(  Rset
    AND   Y
    )
    ST    Y
    RET
  END_FUNCTION_BLOCK


  program FFFB92
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  var
    FF: FFSR
    RSet AT %ix0.0: BOOL
    Set  AT %ix0.1: BOOL
    Out  AT %qx0.0: BOOL
  end_var
    LD  Set
    ST  FF.Set
    LD  RSet
    ST  FF.RSet

    CAL FF
```

```
    LD  FF.Y
    ST  Out
  end_program
```

Übung 9.7 (FFFB93)

```
  (* FUNCTION BLOCK *)
  FUNCTION_BLOCK FFRS
  VAR_INPUT
    Set  : BOOL
    Rset : BOOL
  END_VAR
  VAR_OUTPUT
    Y : BOOL
  END_VAR
    LDN  RSet
    AND( Set
    OR   Y
    )
    ST   Y
    RET
  END_FUNCTION_BLOCK
```

Übung 9.8 (TANK91)

```
  FUNCTION_BLOCK Tank
  VAR_INPUT
    LIS1 : BOOL
    LIS2 : BOOL
    Start: BOOL
  END_VAR
  VAR_OUTPUT
    V1 :   BOOL
    V3 :   BOOL
  END_VAR
  VAR
    V1FF  : RS
    V3FF  : RS
  END_VAR
    LD    Start
    ANDN  V3
    ANDN  LIS2
    ST    V1FF.S
    LD    LIS2
```

```
  ST    V1FF.R1
  ST    V3FF.S
  LDN   LIS1
  ST    V3FF.R1
  cal   V1FF
  cal   V3FF
  ld    V3FF.Q1
  st    V3
  ld    V1FF.Q1
  st    V1
END_FUNCTION_BLOCK

program OneTank
(*Processes:                *)
(*  Tanks (small)           *)
var
(* Instanziierung *)
  Tank0: Tank
  LIS01 AT %IX0.1: bool
  LIS02 AT %IX0.2: bool
  V01   AT %QX0.1: bool
  V03   AT %QX0.3: bool
  Start AT %IX0.7: bool
end_var

(* Verknuepfung der Eingangsklemmen *)
   ld Start
   st Tank0.Start
   ld LIS01
   st Tank0.LIS1
   ld LIS02
   st Tank0.LIS2

(* Funktionsbausteinaufruf *)
   cal Tank0

(* Verknuepfung der Ausgangsklemmen *)
   ld Tank0.V1
   st V01
   ld Tank0.V3
   st V03

end_program
```

Übung 9.9 (TANK92)

```
program ThreeTanks
(*Processes:                *)
(*  Tanks (big)             *)
var
(* Instanziierung *)
  Tank0: Tank
  LIS01 AT %IX0.1: bool
  LIS02 AT %IX0.2: bool
  V01   AT %QX0.1: bool
  V03   AT %QX0.3: bool
  Tank1: Tank
  LIS11 AT %IX1.1: bool
  LIS12 AT %IX1.2: bool
  V11   AT %QX1.1: bool
  V13   AT %QX1.3: bool
  Tank2: Tank
  LIS21 AT %IX2.1: bool
  LIS22 AT %IX2.2: bool
  V21   AT %QX2.1: bool
  V23   AT %QX2.3: bool
  Start AT %IX0.7: bool
end_var


(* Verknuepfung der Eingangsklemmen *)
   ld Start
   st Tank0.Start
   ld LIS01
   st Tank0.LIS1
   ld LIS02
   st Tank0.LIS2
(* Funktionsbausteinaufruf *)
   cal Tank0

   ld Start
   st Tank1.Start
   ld LIS11
   st Tank1.LIS1
   ld LIS12
   st Tank1.LIS2
(* Funktionsbausteinaufruf *)
   cal Tank1

   ld Start
```

```
   st Tank2.Start
   ld LIS21
   st Tank2.LIS1
   ld LIS22
   st Tank2.LIS2
(* Funktionsbausteinaufruf *)
   cal Tank2


(* Verknuepfung der Ausgangsklemmen *)
   ld Tank0.V1
   st V01
   ld Tank0.V3
   st V03
   ld Tank1.V1
   st V11
   ld Tank1.V3
   st V13
   ld Tank2.V1
   st V21
   ld Tank2.V3
   st V23

end_program
```

Kapitel 10
Sprünge, Schleifen und Wiederholungen

Übung 10.1 (FLASH101)

```
Program flash101
(*Processes:                 *)
(*  Standard-I/O  byte 1     *)
(*  Hex-Output    byte 1     *)
var
  Pulse: TON
  Cycle1: bool
  Bit0 AT %QX1.0: bool
  Byte1 AT %QB1: usint
end_var
(* set Bit *)
  ldn Cycle1
  s Cycle1
  s Bit0

(* Pulsgenerator *)
  ldn Pulse.Q
  st  Pulse.IN
  ld  t#1000ms
  st  Pulse.PT
  cal Pulse

(* Go!! *)
  ldn Pulse.Q
  jmpc PulseOK
  ld Byte1
  mul 2
  st Byte1
PulseOK:
end_program
```

Übung 10.2 (FLASH102)

```
Program FlashLightFlash102
(*Processes:                 *)
(*  Standard-I/O  byte 1     *)
(*  Hex-Output    byte 1     *)
var
  Pulse: TON
  Cycle1: bool
  Bit0 AT %QX1.0: bool
  Bit7 AT %QX1.7: bool
  Byte1 AT %QB1: usint
end_var
(* set Bit *)
  ldn Cycle1
  s Cycle1
  s Bit0

(* Pulsgenerator *)
  ldn Pulse.Q
  st  Pulse.IN
  ld  t#1000ms
  st  Pulse.PT
  cal Pulse

(* Go!! *)
  ldn Pulse.Q
  jmpc PulseOK

  ld Byte1
  eq 128
  r Bit7
  r Cycle1

  ld Byte1
  mul 2
  st Byte1
PulseOK:
end_program
```

Übung 10.3 (FLASH103)

```
Program Flash103
(*Processes:                 *)
(*  Standard-I/O  byte 1     *)
```

```
(*  Hex-Output    byte 1    *)
var
  Pulse: TON
  Cycle1: bool
  Bit7 AT %QX1.7: bool
  Byte1 AT %QB1: usint
end_var
(* set Bit *)
  ldn Cycle1
  s Cycle1
  s Bit7

(* Pulsgenerator *)
  ldn Pulse.Q
  st  Pulse.IN
  ld  t#1000ms
  st  Pulse.PT
  cal Pulse

(* Go!! *)
  ldn Pulse.Q
  jmpc PulseOK
  ld Byte1
  div 2
  st Byte1
  eq 0
  s  Bit7
PulseOK:
end_program
```

Übung 10.4 (FLASH104)

```
Program Flash104
(*Processes:                *)
(*  Standard-I/O  byte 1    *)
(*  Hex-Output    byte 1    *)
var
  Pulse: TON
  Cycle1: bool
  Bit1 AT %QX1.0: bool
  Bit7 AT %QX1.7: bool
  Byte1 AT %QB1: usint
  GoLeft AT %IX1.0: BOOL
end_var
(* set Bit *)
```

```
    ldn Cycle1
    s Cycle1
    s Bit1

  (* Pulsgenerator *)
    ldn Pulse.Q
    st  Pulse.IN
    ld  t#1000ms
    st  Pulse.PT
    cal Pulse

  (* Go!! *)
    ldn Pulse.Q
    jmpc PulseOK
    ld   GoLeft
    jmpcn RunRight
  RunLeft:
    ld Byte1
    eq 128
    r Bit7
    r Cycle1

    ld Byte1
    mul 2
    st Byte1
    jmp PulseOK
  RunRight:
    ld Byte1
    div 2
    st Byte1
    eq 0
    s  Bit7
  PulseOK:
  end_program
```

Übung 10.5 Keine Musterlösung.

Übung 10.6 (TANK102)

```
  program Tank102_ThreeTanks
  (*Processes:                *)
  (*  Tanks (big)             *)
  var
  (* Instanziierung *)
    Tank0: Tank
```

```
    LIS01 AT %IX0.1: bool
    LIS02 AT %IX0.2: bool
    V01   AT %QX0.1: bool
    V03   AT %QX0.3: bool
    Tank1: Tank
    LIS11 AT %IX1.1: bool
    LIS12 AT %IX1.2: bool
    V11   AT %QX1.1: bool
    V13   AT %QX1.3: bool
    Tank2: Tank
    LIS21 AT %IX2.1: bool
    LIS22 AT %IX2.2: bool
    V21   AT %QX2.1: bool
    V23   AT %QX2.3: bool
    Start AT %IX0.7: bool
  end_var


  (* Verknuepfung der Eingangsklemmen *)
     ld Start
     st Tank0.Start
     ld LIS01
     st Tank0.LIS1
     ld LIS02
     st Tank0.LIS2
  (* Funktionsbausteinaufruf *)
     cal Tank0

     ld Tank0.LIS2
     st Tank1.Start
     ld LIS11
     st Tank1.LIS1
     ld LIS12
     st Tank1.LIS2
  (* Funktionsbausteinaufruf *)
     cal Tank1

     ld Tank1.LIS2
     st Tank2.Start
     ld LIS21
     st Tank2.LIS1
     ld LIS22
     st Tank2.LIS2
  (* Funktionsbausteinaufruf *)
     cal Tank2
```

```
(* Verknuepfung der Ausgangsklemmen *)
   ld Tank0.V1
   st V01
   ld Tank0.V3
   st V03
   ld Tank1.V1
   st V11
   ld Tank1.V3
   st V13
   ld Tank2.V1
   st V21
   ld Tank2.V3
   st V23

end_program
```

Übung 10.7 (7SEG101)

```
Program SevenSegment
(*Processes:                *)
(*  Standard-I/O  byte 1    *)
(*  7-Segment               *)
var
  InByte  AT %IB1: Byte
  OutByte AT %QB0: Byte
end_var
  ld InByte
  st OutByte
end_program
```

Übung 10.8 (7SEG102)

```
Program SevenSegment
(*Processes:                *)
(*  Standard-I/O  byte 1    *)
(*  Hex-Input     byte 1    *)
(*  7-Segment               *)
Var
  InValue AT %IB1: sint
  OutByte AT %QB0: Byte
  Digit: sint
end_var
  ld InValue
```

```
st Digit
eq 0
jmpc Zero
ld Digit
eq 1
jmpc One
ld Digit
eq 2
jmpc Two
ld Digit
eq 3
jmpc Three
ld Digit
eq 4
jmpc Four
ld Digit
eq 5
jmpc Five
ld Digit
eq 6
jmpc Six
ld Digit
eq 7
jmpc Seven
ld Digit
eq 8
jmpc Eight
ld Digit
eq 9
jmpc Nine
ld Digit
eq 16#a
jmpc _A
ld Digit
eq 16#b
jmpc _B
ld Digit
eq 16#c
jmpc _C
ld Digit
eq 16#d
jmpc _D
ld Digit
eq 16#e
jmpc _E
```

```
  ld Digit
  eq 16#f
  jmpc _F

Zero:
  ld 2#00111111
  jmp End
One:
  ld 2#00000110
  jmp End
Two:
  ld 2#01011011
  jmp End
Three:
  ld 2#01001111
  jmp End
Four:
  ld 2#01100110
  jmp End
Five:
  ld 2#01101101
  jmp End
Six:
  ld 2#01111101
  jmp End
Seven:
  ld 2#00000111
  jmp End
Eight:
  ld 2#01111111
  jmp End
Nine:
  ld 2#01101111
  jmp End
_A:
  ld 2#01110111
  jmp End
_B:
  ld 2#01111100
  jmp End
_C:
  ld 2#00111001
  jmp End
_D:
  ld 2#01011110
```

```
    jmp End
  _E:
    ld 2#01111001
    jmp End
  _F:
    ld 2#01110001
    jmp End

  End:
    st OutByte
  end_program
```

Übung 10.9 (DICE101)

```
Function_Block Chance16
var_input
  run: bool
end_var
var_output
  Value: sint
end_var
var
  Counter: CTU
  toggle: bool
end_var

  ldn  run
  jmpc ausgabe

  ldn toggle
  st  toggle
  st  Counter.CU
  cal Counter

  ld  Counter.cv
  gt  5
  st  Counter.r
  cal Counter
ausgabe:
  ld  Counter.CV
  int_to_sint
  add 1
  st  Value
end_function_block
```

```
 Program ProgDice101
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Hex-Output              *)
 var
  Start AT %ix0.0: BOOL
  OutByte AT %qb0: sint
  Dice: Chance16
end_var
  ld Start
  st Dice.run
  cal Dice
  ld Dice.Value
  st OutByte
end_program
```

Kapitel 11
Funktionen

Übung 11.1 Keine Musterlösung.

Übung 11.2 (7SEG111)

```
Function int_to_7seg : Byte
var_input
  Digit: int
end_var
  ld Digit
  eq 0
  jmpc Zero
  ld Digit
  eq 1
  jmpc One
  ld Digit
  eq 2
  jmpc Two
  ld Digit
  eq 3
  jmpc Three
  ld Digit
  eq 4
  jmpc Four
  ld Digit
  eq 5
  jmpc Five
  ld Digit
  eq 6
  jmpc Six
  ld Digit
  eq 7
  jmpc Seven
```

```
    ld Digit
    eq 8
    jmpc Eight
    ld Digit
    eq 9
    jmpc Nine
    ld Digit
    eq 16#a
    jmpc _A
    ld Digit
    eq 16#b
    jmpc _B
    ld Digit
    eq 16#c
    jmpc _C
    ld Digit
    eq 16#d
    jmpc _D
    ld Digit
    eq 16#e
    jmpc _E
    ld Digit
    eq 16#f
    jmpc _F

  Zero:
    ld 2#00111111
    jmp End
  One:
    ld 2#00000110
    jmp End
  Two:
    ld 2#01011011
    jmp End
  Three:
    ld 2#01001111
    jmp End
  Four:
    ld 2#01100110
    jmp End
  Five:
    ld 2#01101101
    jmp End
  Six:
    ld 2#01111101
```

```
    jmp End
  Seven:
    ld 2#00000111
    jmp End
  Eight:
    ld 2#01111111
    jmp End
  Nine:
    ld 2#01101111
    jmp End
  _A:
    ld 2#01110111
    jmp End
  _B:
    ld 2#01111100
    jmp End
  _C:
    ld 2#00111001
    jmp End
  _D:
    ld 2#01011110
    jmp End
  _E:
    ld 2#01111001
    jmp End
  _F:
    ld 2#01110001
    jmp End

  End:
    st int_to_7seg
    ret

  end_function
```

____

```
  Program SevenSegment111
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  (*  Hex-Input     byte 0    *)
  (*  7-Segment     byte 0    *)
  var
    InByte0  AT %IB0: byte
    OutByte0 AT %QB0: byte
  end_var
```

```
    ld InByte0
    byte_to_int
    int_to_7seg
    st OutByte0
  end_program
```

Übung 11.3 (DICE111)

```
  Function int_to_dice : Byte
  var_input
    Digit: int
  end_var
    ld Digit
    eq 1
    jmpc One
    ld Digit
    eq 2
    jmpc Two
    ld Digit
    eq 3
    jmpc Three
    ld Digit
    eq 4
    jmpc Four
    ld Digit
    eq 5
    jmpc Five
    ld Digit
    eq 6
    jmpc Six
    jmp Fault
  One:
    ld 2#01000000
    jmp End
  Two:
    ld 2#00001001
    jmp End
  Three:
    ld 2#01100100
    jmp End
  Four:
    ld 2#00101101
    jmp End
  Five:
    ld 2#01101101
```

```
    jmp End
  Six:
    ld 2#00111111
    jmp End
  Fault:
    ld 0
  End:
    st int_to_dice
    ret
  end_function
```

———

```
  Program DiceProg111
  (*Processes:                *)
  (*  Panel                   *)
  (*  Dice          byte 1    *)
  var
    Start AT %ix0.0: BOOL
    OutValue AT %qb3: sint
    OutDice AT %qb1: BYTE
    Dice: Chance16
  end_var
    ld Start
    st Dice.run
    cal Dice
    ld Dice.Value
    st OutValue
    int_to_dice
    st OutDice
  end_program
```

Übung 11.4 (BCD111)

```
  (* FUNCTION *)
  Function hex_to_bcd: Byte
  var_input
    Digit: int
  end_var
  var
    BCDDigitLow:  int
    BCDDigitHigh: int
    BCDByteLow: byte
  end_var
    ld   0
    st   BCDDigitHigh
```

```
  ld   Digit
  st   BCDDigitLow
gt100:
  lt   100
  jmpc lt100
  ld   BCDDigitLow
  sub  100
  st   BCDDigitLow
  jmp  gt100
lt100:
  ld   BCDDigitLow
  lt   10
  jmpc lt10
  ld   BCDDigitLow
  sub  10
  st   BCDDigitLow
  ld   BCDDigitHigh
  add  1
  st   BCDDigitHigh
  ld   BCDDigitLow
  jmp  lt100
lt10:
  ld   BCDDigitLow
  int_to_byte
  st   BCDByteLow
  ld   BCDDigitHigh
  mul  16
  int_to_byte
  or   BCDByteLow
  st   hex_to_bcd
  ret
end_function
```

———-

```
Program Mengenmessung
(*Processes:                  *)
(* Tank (small)               *)
(* Hex-Output      byte 2     *)
var
  Switch AT %ix0.0: BOOL
  Start AT %ix0.7: BOOL
  Stop  AT %ix0.6: BOOL
  LIS1  AT %ix0.1: BOOL
  LIS2  AT %ix0.2: BOOL
  V1    AT %qx0.1: BOOL
```

```
    V3    AT %qx0.3: BOOL
    Display AT %qb3: BYTE
    HexDisplay AT %qb2: BYTE
    Zaehler: CTU
    Pulse: TON
    count: bool
    run: bool
  end_var
  (* Counter *)
    ld Pulse.Q
    st Zaehler.CU

  (* Start   *)
    ld LIS1
    and run
    s count
    ld Start
    s run
  (* Stopp   *)
    ld LIS2
    r count
    ld Stop
    r run
  (* Counter Reset *)
    ld Stop
    r run
  //  st Zaehler.R
    r count
  (* Fill   *)
    ld run
    st V1
  (* Leeren *)
    ld Switch
    st V3
  (* Generator *)
    ldn Pulse.Q
    and count   (* Lauf *)
    st Pulse.IN
    ld 60
    st Pulse.PT

    cal Zaehler
    cal Pulse

    ld Zaehler.CV
```

```
    hex_to_bcd
    st Display
    ld Zaehler.CV
    int_to_byte
    st HexDisplay
  end_program
```

Übung 11.5 (MAX111)
Beachten Sie bei dieser Aufgabe, dass die Funktion mit INT-Werten arbeitet. Diese sind 2 Bytes groß; daher werden für die Eingaben zwei Elemente benötigt, nämlich z.B. `%ib0` und `%ib1` für das „Digit1“ und `%ib2` und `%ib3` für das „Digit2“. Weil das oberste Bit das Vorzeichen darstellt, ist „3500“ größer als „8112“!! (siehe Abb. 11.1) Sie können das auch durch Verwendung der Typen sint und usind testen. (siehe Abb. 11.2)

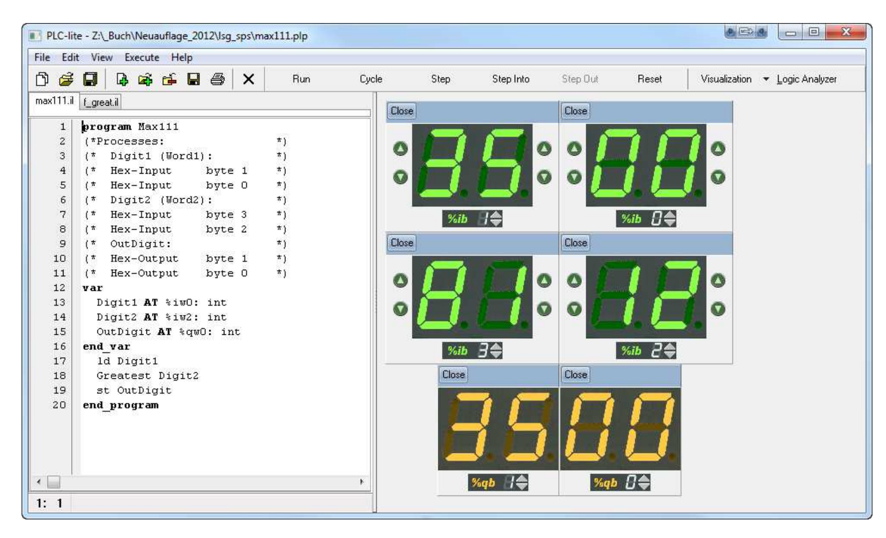


Abb. 11.1 zu Übung 11.5 (MAX111)

```
  (* FUNCTION *)
  Function Greatest: int
  var_input
    D1: int
    D2: int
  end_var
    ld D1
    gt D2
    jmpc D1Great
```

```
D2Great:
  ld D2
  st Greatest
  ret
D1Great:
  ld D1
  st Greatest
  ret
end_function
```

Abb. 11.2 zu Übung 11.5 (MAX111)

```
(* PROGRAM *)
program Max111
(*Processes:                    *)
(*  Digit1 (Word1):             *)
(*  Hex-Input      byte 1       *)
(*  Hex-Input      byte 0       *)
(*  Digit2 (Word2):             *)
(*  Hex-Input      byte 3       *)
(*  Hex-Input      byte 2       *)
(*  OutDigit:                   *)
(*  Hex-Output     byte 1       *)
(*  Hex-Output     byte 0       *)
var
  Digit1 AT %iw0: int
  Digit2 AT %iw2: int
```

```
  OutDigit AT %qw0: int
end_var
  ld Digit1
  Greatest Digit2
  st OutDigit
end_program
```

Übung 11.6 (TIME111)

```
(* FUNCTION *)
FUNCTION_BLOCK SEC01
VAR
  Pulse: TON
END_VAR
VAR_OUTPUT
  Y: Bool
END_VAR
(* Impulsgenerator *)
  ldn Pulse.Q
  st  Pulse.IN
  ld  t#100ms
  st  Pulse.PT
  cal Pulse
  ld  Pulse.Q
  st  Y
END_FUNCTION_BLOCK
```

———

```
(* PROGRAM *)
program winner
(*Standard-I/O*)
(*%i0.0 reset and start     *)
(*%i0.1 , %i0.2 stopps and  *)
(*shows actual Time1 / Time2*)
(*processes:                *)
(*      Standard-I/O byte 0 *)
(* act  HEX-Output   byte 1 *)
(* min  HEX-Output   byte 2 *)
(* max  HEX-Output   byte 3 *)
var
  StartReset AT %ix0.0: BOOL
  Stop1 AT %ix0.1: BOOL
  Stop2 AT %ix0.2: BOOL
  Display AT %qb1: Byte
  DisplayMin AT %qb2: Byte
```

```
    DisplayMax AT %qb3: Byte
    Lamp1 AT %qx0.1: BOOL
    Lamp2 AT %qx0.2: BOOL
    c1: ctu
    c2: ctu
    Ticker: sec01
    run1: bool
    run2: bool
  end_var
    ld StartReset
    s run1
    s run2
    st c1.r
    st c2.r
    ld Stop1
    r run1
    ld Stop2
    r run2
    ld Ticker.Y
    and run1
    st c1.cu
    ld Ticker.Y
    and run2
    st c2.cu

    cal Ticker
    cal c1
    cal c2

    ld Stop2
    jmpc Display2
    ld Stop1
    jmpc Display1

  Display:
    ld c1.cv
    Greatest c2.cv
    int_to_byte
    st Display
  MaxDisplay:
    ld c1.cv
    Greatest c2.cv
    int_to_byte
    st DisplayMax
  MinDisplay:
```

```
    ld c1.cv
    Least c2.cv
    int_to_byte
    st DisplayMin
    jmp End

  Display2:
    ld c2.cv
    int_to_byte
    st Display
    jmp End
  Display1:
    ld c1.cv
    int_to_byte
    st Display

  End:
    ld run1
    or Stop1
    st Lamp1
    ld run2
    or Stop2
    st Lamp2

  end_program
```

Übung 11.7 (DISKR111)

```
  (* FUNCTION *)
  Function Between: bool
  var_input
    z1: sint
    z2: sint
    z3: sint
  end_var
    ld z2
    ge z3
    jmpc z32
    ld z2
    lt z3
    jmpc z23
    jmp  NotBetween
  z23:
    ld z1
    lt z2
```

```
    jmpc NotBetween
    ld z1
    gt z3
    jmpc NotBetween
    jmp  IsBetween
  z32:
    ld z1
    lt z3
    jmpc NotBetween
    ld z1
    gt z2
    jmpc NotBetween
    jmp  IsBetween
  NotBetween:
    ld   0
    st   Between
    ret
  IsBetween:
    ld   1
    st   Between
    ret
  end_function
```

```
  (* PROGRAM *)
  program Diskriminator
  (*      Standard-I/O byte 0 *)
  (*      HEX-Input    byte 0 *)
  (* min  HEX-Input    byte 1 *)
  (* max  HEX-Input    byte 2 *)
  var
    CompValue AT %ib0: sint
    MinValue  AT %ib1: sint
    MaxValue  AT %ib2: sint
    OKLamp AT %qx0.0: BOOL
  end_var
    ld Compvalue
    Between MinValue, MaxValue
    st OKLamp
  end_program
```

Kapitel 12
Ablaufsteuerungen

Übung 12.1 (PUSH121)

```
Program PushButton1
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
VAR
  Button AT %ix0.0: BOOL
  Lamp   AT %qx0.0: BOOL
  Step1: bool
  Step2: bool
  Cycle1:bool
END_VAR
  ldn Cycle1
  s   Cycle1
  s   Step1

(* Step 1 *)
  LD   Step1    (*Step 1 activ?     *)
  AND  Button   (*Button pushed?   *)
  R    Step1    (*Step 1 leave      *)
  S    Step2    (*Step 2 activate   *)
(* Step 2 *)
  LD   Step2    (*Step 2 activ?     *)
  ANDN Button   (*Button released?  *)
  R    Step2    (*Step 2 leave      *)
  S    Step1    (*Step 1 activate   *)
(* ACTION   *)
  LD   Step2    (*Step 2 activ?     *)
  ST   Lamp     (*Lamp on          *)
end_program
```

Übung 12.2 (PUSH122)

```
Program PushButton2
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Standard-I/O  byte 1    *)
VAR
  Step1 AT %qx1.0: bool
  Step2 AT %qx1.1: bool
  Step3 AT %qx1.2: bool
  Step4 AT %qx1.3: bool
  Cycle1:bool
  Button AT %IX0.0: bool
  Lamp   AT %QX0.0: bool
END_VAR

  ldn Cycle1
  s   Cycle1
  s   Step1
(* Step 1 *)
  LD   Step1   (* Step 1 aktiv?     *)
  AND  Button  (* Button pressed?   *)
  R    Step1   (* Step 1 verlassen  *)
  S    Step2   (* Step 2 aktivieren *)
(* Step 2 *)
  LD   Step2   (* Step 2 aktiv?     *)
  ANDN Button  (* Button released?  *)
  R    Step2   (* Step 2 verlassen  *)
  S    Step3   (* Step 3 aktivieren *)
(* Step 3 *)
  LD   Step3   (* Step 3 aktiv?     *)
  AND  Button  (* Button pressed?   *)
  R    Step3   (* Step 3 verlassen  *)
  S    Step4   (* Step 4 aktivieren *)
(* Step 4 *)
  LD   Step4   (* Step 4 aktiv?     *)
  ANDN Button  (* Button released?  *)
  R    Step4   (* Step 4 verlassen  *)
  S    Step1   (* Step 1 aktivieren *)
(* ACTION   *)
  LD   Step2   (* Step 2 aktiv?     *)
  OR   Step3   (* Step 3 aktiv?     *)
  ST   Lamp    (* Lamp on           *)
end_program
```

Übung 12.3 (PUSH123)

```
Program PushButton3
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Standard-I/O  byte 1    *)
VAR
  Step1 AT %qx1.0: bool
  Step2 AT %qx1.1: bool
  Step3 AT %qx1.2: bool
  Step4 AT %qx1.3: bool
  Cycle1:bool
  Button AT %IX0.0: bool
  Lamp   AT %QX0.0: bool
END_VAR
  ldn Cycle1
  s   Cycle1
  s   Step1
(* Ablauf der Schrittkette *)
(* Step 1 *)
  LD   Step1
  AND  Button  (* Button pressed?  *)
  R    Step1
  S    Step2
(* Step 2 *)
  LD   Step2
  ANDN Button  (* Button released? *)
  R    Step2
  S    Step3
(* Step 3 *)
  LD   Step3
  AND  Button  (* Button pressed?  *)
  R    Step3
  S    Step4
(* Step 4 *)
  LD   Step4
  ANDN Button  (* Button released? *)
  R    Step4
  S    Step1
(* ACTION   *)
  LD   Step2   (* Step 2 activ     *)
  S    Lamp    (* Lamp on          *)
  LD   Step4   (* Step 4 activ?    *)
  R    Lamp    (* Lamp off         *)
end_program
```

Übung 12.4 (TANK121)

```
Program Tank121
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Standard-I/O  byte 1    *)
VAR
  V1 AT %qx0.1: BOOL
  V3 AT %qx0.3: BOOL
  LIS1 AT %ix0.1: BOOL
  LIS2 AT %ix0.2: BOOL
  Start AT %ix0.7: BOOL
  S01 AT %qx1.0:bool
  S02 AT %qx1.1:bool
  S03 AT %qx1.2:bool
  Cycle1:bool
END_VAR
  ldn Cycle1
  s   Cycle1
  s   S03
(* Ablauf der Schrittkette *)
(* STEP 1 *)
  LD   S01   (*Step 1 activ?   *)
  AND  LIS2  (*full?           *)
  R    S01   (*Step 1 leave    *)
  S    S02   (*Step 2 activate *)
(* STEP 2 *)
  LD   S02   (*Step 2 activ?   *)
  ANDN LIS1  (*empty?          *)
  R    S02   (*Step 2 leave    *)
  S    S03   (*Step 3 activate *)
(* STEP 3 *)
  LD   S03   (*Step 3 activ?   *)
  AND  Start (*Start pushed?   *)
  R    S03   (*Step 3 leave    *)
  S    S01   (*Step 1 activate *)
(* ACTION *)
  LD   S01   (*Step 1 activ?   *)
  ST   V1    (*V1 open         *)
  LD   S02   (*Step 2 activ?   *)
  ST   V3    (*V3 open         *)
end_program
```

Übung 12.5 (TANK122)

```
Program Tank122
```

```
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Standard-I/O  byte 1    *)
VAR
  V1 AT %qx0.1: BOOL
  V3 AT %qx0.3: BOOL
  LIS1 AT %ix0.1: BOOL
  LIS2 AT %ix0.2: BOOL
  Start AT %ix0.7: BOOL
  S01 AT %qx1.0:bool
  S02 AT %qx1.1:bool
  S03 AT %qx1.2:bool
  S04 AT %qx1.3:bool
  Cycle1:bool
END_VAR
  ldn Cycle1
  s   Cycle1
  s   S03
(* Ablauf der Schrittkette *)
(* STEP 1 *)
  LD   S01   (*Step 1 activ?   *)
  AND  LIS2  (*full?           *)
  R    S01   (*Step 1 leave    *)
  S    S02   (*Step 2 activate *)
(* STEP 2 *)
  LD   S02   (*Step 2 activ?   *)
  ANDN LIS1  (*empty?          *)
  R    S02   (*Step 2 leave    *)
  S    S03   (*Step 3 activate *)
(* STEP 3 *)
  LD   S03   (*Step 3 activ?   *)
  ANDN Start (*Start  released?*)
  R    S03   (*Step 3 leave    *)
  S    S04   (*Step 4 activate *)
(* STEP 4 *)
  LD   S04   (*Step 4 activ?   *)
  AND  Start (*Start  pressed  *)
  R    S04   (*Step 4 leave    *)
  S    S01   (*Step 1 activate *)
(* ACTION *)
  LD   S01   (*Step 1 activ?   *)
  ST   V1    (*V1 open         *)
  LD   S02   (*Step 2 activ?   *)
  ST   V3    (*V3 open         *)
end_program
```

Übung 12.6 (MIXER121)

```
Program TankAndMixer121
(*Processes:                *)
(*  Chemical Process (small)*)
VAR
  V1 AT %qx0.1: BOOL
  V3 AT %qx0.3: BOOL
  LIS1 AT %ix0.1: BOOL
  LIS2 AT %ix0.2: BOOL
  Start AT %ix0.7: BOOL
  Stop  AT %ix0.6: BOOL
  Lamp  AT %qx0.0: BOOL
  Mixer AT %qx0.6: BOOL
  Heater AT %qx0.5: BOOL
  TIC50  AT %ix0.5: BOOL
  V2     AT %qx0.2: BOOL
  V4     AT %qx0.4: BOOL
  LIS3   AT %ix0.3: BOOL
  S01:bool
  S02:bool
  S03:bool
  S04:bool
  S05:bool
  S06:bool
  S06a:bool
  Cycle1:bool
END_VAR
  ldn Cycle1
  s   Cycle1
  s   S06a
(* STEP 1 *)
  LD   S01
  AND  LIS2  (*full?           *)
  R    S01
  S    S02
(* STEP 2 *)
  LD   S02
  ANDN LIS1  (*empty?          *)
  R    S02
  S    S03
(* STEP 3 *)
  LD   S03
  AND  TIC50 (*Temp. OK? *)
  R    S03
  S    S04
```

```
(* STEP 4 *)
  LD   S04
  AND  Start (*Start pressed?         *)
  R    S04
  S    S05
(* STEP 5 *)
  LD   S05
  ANDN LIS3  (*empty?          *)
  R    S05
  S    S06
(* STEP 6 *)
  LD   S06
  ANDN Start (*Start released?*)
  R    S06
  S    S06a
(* STEP 6a *)
  LD   S06a
  AND  Start (*Start pressed? *)
  R    S06a
  S    S01
(* ACTION *)
  LD   S01   (*Step 1 activ?   *)
  ST   V1
  LD   S02   (*Step 2 activ?   *)
  ST   V3
  LD   S03   (*Step 3 activ?   *)
  ST   Heater
  S    Mixer
  S    V2
  LD   S05   (*Step 5 activ?   *)
  ST   V4
  R    Mixer
  LD   S06   (*Step 6 activ?   *)
  R    V2
end_program
```

Übung 12.7 (MIXER122)

```
Program TankAndMixer122
(*Processes:                *)
(*  Chemical Process (small)*)
VAR
  V1 AT %qx0.1: BOOL
  V3 AT %qx0.3: BOOL
  LIS1 AT %ix0.1: BOOL
```

```
    LIS2 AT %ix0.2: BOOL
    Start AT %ix0.7: BOOL
    Mixer AT %qx0.6: BOOL
    Heater AT %qx0.5: BOOL
    TIC50  AT %ix0.5: BOOL
    V2     AT %qx0.2: BOOL
    V4     AT %qx0.4: BOOL
    LIS3   AT %ix0.3: BOOL
    S01:bool
    S02:bool
    S03:bool
    S04:bool
    S05:bool
    S06:bool
    S06a:bool
    Cycle1:bool
    TimeMix: TON
  END_VAR
    ldn Cycle1
    s   Cycle1
    s   S06a
  (* STEP 1 *)
    LD   S01
    AND  LIS2  (*full?           *)
    R    S01
    S    S02
  (* STEP 2 *)
    LD   S02
    ANDN LIS1  (*empty?          *)
    R    S02
    S    S03
  (* STEP 3 *)
    LD   S03
    AND  TIC50 (*Temp. OK? *)
    R    S03
    S    S04
  (* STEP 4 *)
    LD   S04
    AND  TimeMix.Q (*Time finished? *)
    R    S04
    S    S05
  (* STEP 5 *)
    LD   S05
    ANDN LIS3  (*empty?          *)
    R    S05
```

```
  S    S06
(* STEP 6 *)
  LD   S06
  ANDN Start (*Start released?*)
  R    S06
  S    S06a
(* STEP 6a *)
  LD   S06a
  AND  Start (*Start pressed? *)
  R    S06a
  S    S01
(* ACTION *)
  LD   S01   (*Step 1 activ?   *)
  ST   V1
  LD   S02   (*Step 2 activ?   *)
  ST   V3
  LD   S03   (*Step 3 activ?   *)
  ST   Heater
  S    Mixer
  S    V2
  LD   S04   (*Step 4 activ?   *)
  ST   TimeMix.IN
  CAL  TimeMix
  LD   S05   (*Step 5 activ?   *)
  ST   V4
  R    Mixer
  LD   S06a  (*Step 6a activ?   *)
  R    V2
(* Timer *)
  LD   t#10s
  ST   TimeMix.PT
end_program
```

Kapitel 13
Wiederholungsaufgaben

Übung 13.1 (LOGIK27)
Siehe Abb. 13.1 und 13.2 ◇

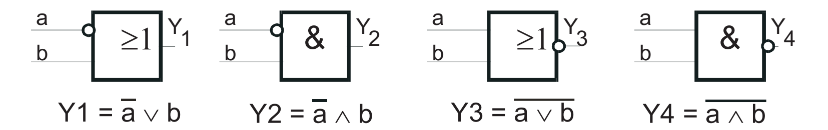


Abb. 13.1 Schaltglieder zu Übung 13.1

a	b	Y	Y1	Y2	Y3	Y4
0	0	1	1	0	1	1
1	0	0	0	0	0	1
0	1	0	1	1	0	1
1	1	0	1	0	0	0

Abb. 13.2 Funktionstabelle zu Übung 13.1

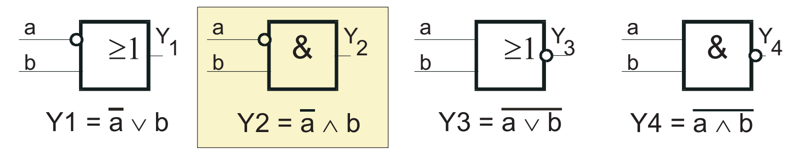
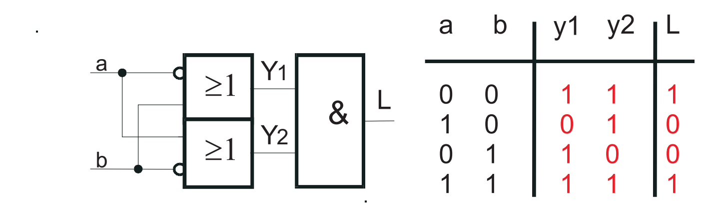


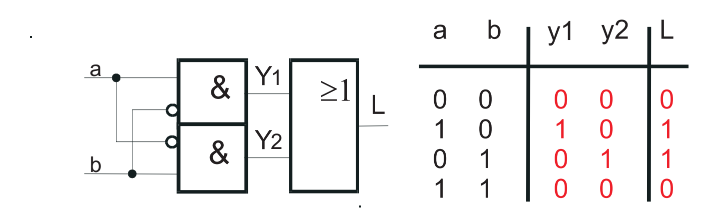
Abb. 13.3 Schaltglied zu Übung 13.2

Übung 13.2 (LOGIK28)
Siehe Abb. 13.3 ◇



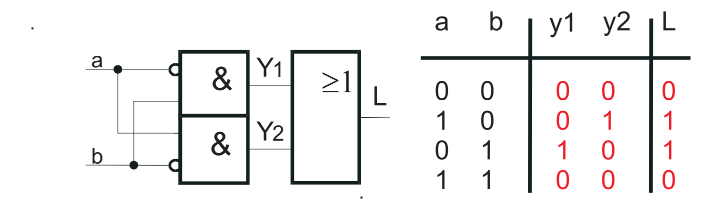
a	b	y1	y2	L
0	0	1	1	1
1	0	0	1	0
0	1	1	0	0
1	1	1	1	1

Abb. 13.4 Funktion 1 zu Übung 13.3



a	b	y1	y2	L
0	0	0	0	0
1	0	1	0	1
0	1	0	1	1
1	1	0	0	0

Abb. 13.5 Funktion 2 zu Übung 13.3



a	b	y1	y2	L
0	0	0	0	0
1	0	0	1	1
0	1	1	0	1
1	1	0	0	0

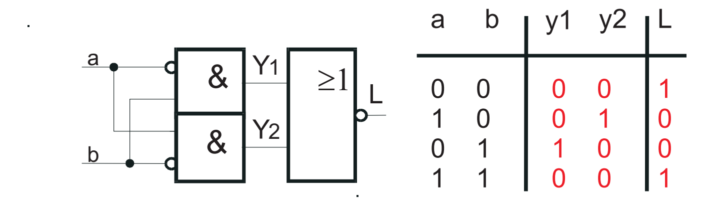
Abb. 13.6 Funktion 3 zu Übung 13.3

Übung 13.3 (LOGIK29)
Siehe Abb. 13.4 bis Abb. 13.7 ◇

Übung 13.4 (LOGIK30)
Siehe Abb. 13.8 ◇

Übung 13.5 (FLASH23)
Siehe Abb. 13.9

◇



a	b	y1	y2	L
0	0	0	0	1
1	0	0	1	0
0	1	1	0	0
1	1	0	0	1

Abb. 13.7 Funktion 4 zu Übung 13.3

a	b	c	d	e	X1	X2	X
1	0	0	0	0	0	1	0
1	1	0	0	0	1	1	1
1	0	1	0	0	1	1	1
1	1	1	0	0	1	1	1
1	0	0	1	0	0	1	0
1	1	0	1	0	1	1	1
1	0	1	1	0	1	1	1
1	1	1	1	0	1	1	1
1	0	0	0	1	0	1	0
1	1	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	1	0	1	1	1	1
1	0	0	1	1	0	0	0
1	1	0	1	1	1	0	0
1	0	1	1	1	1	0	0
1	1	1	1	1	1	0	0

Abb. 13.8 Schaltung zu Übung 13.4



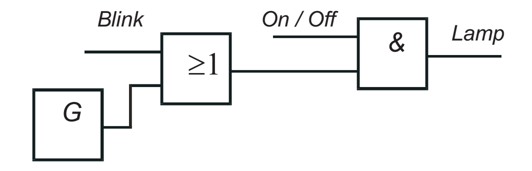


Abb. 13.9 Schaltung zu Übung 13.5 (FLASH23)

Übung 13.6 (TEMP51)

```
PROGRAM Temp51
(*Processes:                  *)
(*  Boiler                    *)
VAR
  Sw1  AT %IX0.0: BOOL
  TIC  AT %IX0.4: BOOL
  Heat AT %QX0.5: BOOL
  Horn AT %QX0.7: BOOL
```

```
END_VAR
  LD   Sw1
  ST   Heat
  LDN  TIC
  ST   Horn
END_PROGRAM
```

◇

Übung 13.7 (LOGIK51)

```
Program Logik51
(*Processes:                *)
(*  Standard I/O  byte0     *)
var
  Form    AT %ix0.0: bool
  Druck   AT %ix0.1: bool
  Gitter  AT %ix0.2: bool
  Temp    AT %ix0.3: bool
  Stempel AT %qx0.0: bool
end_var
  ld   Form
  andn Druck
  and  Gitter
  andn Temp
  st   Stempel
end_program
```

◇

Übung 13.8 (LOGIK52)

```
Program Logik52
(*Processes:                *)
(*  Standard I/O  byte0     *)
var
  Drehzahl AT %ix0.0: bool
  Temp     AT %ix0.1: bool
  Kuehl    AT %ix0.2: bool
  Zulauf   AT %qx0.0: bool
end_var
  ld  Drehzahl
  or  Temp
  or  Kuehl
  stn Zulauf
end_program
```

◇

Übung 13.9 (BOILER53b)
Keine Musterlösung. ◇

Übung 13.10 (BOILER55)

```
Program Boiler55
(*Processes:                *)
(*  Standard I/O  byte0     *)
var
  PressureOK       AT %ix0.0: bool
  TemperatureOK    AT %ix0.1: bool
  ConcentrationOK  AT %ix0.2: bool
  Valve            AT %qx0.0: bool
end_var
  ldn PressureOK
  orn TemperatureOK
  or  ConcentrationOK
  st   Valve
end_program
```

◇

Übung 13.11 (TANK66)

```
Program Tank66
(*Processes:                *)
(*  Tanks (small)           *)
VAR
  V1FF : RS
  V3FF : RS
  Start AT %ix0.7: BOOL
  Stop  AT %ix0.6: BOOL
  LIS1  AT %ix0.1: BOOL
  LIS2  AT %ix0.2: BOOL
  V1    AT %qx0.1: BOOL
  V3    AT %qx0.3: BOOL
  Lamp  AT %qx0.0: BOOL
  Run : BOOL
END_VAR
  ld   Start
  S    Run
  ld   Stop
  R    Run

  ld   Run
  andn V3FF.Q1
  st   V1FF.S
```

```
    ld   LIS2
    st   V1FF.R1

    ld   LIS2
    andn V1FF.Q1
    st   V3FF.S
    ldn  LIS1
    st   V3FF.R1

    cal  V1FF
    cal  V3FF

    ld   V1FF.Q1
    st   V1

    ld   V3FF.Q1
    st   V3

    ld   Run
    st   Lamp
  END_PROGRAM
```

◇

Übung 13.12 (COOLER61)
Keine Musterlösung. ◇

Übung 13.13 (SORTER61)
Keine Musterlösung. ◇

Übung 13.14 (FLASH74)
Keine Musterlösung. ◇

Übung 13.15 (COUNT85)

```
  PROGRAM Count85
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  (*  Hex-Output    byte 1    *)
  VAR
    Counter : CTD
    Disk      AT %IX0.2: Bool
    ChangeBox AT %QX0.2: Bool
    Start     AT %IX0.1: Bool
    CountValue AT %QB1:  INT
  END_VAR
```

```
    LD  10
    ST  Counter.PV
    LD  Disk
    ST  Counter.CD
    LD  Start
    AND ChangeBox
    ST  Counter.LD

    CAL Counter

    LD  Counter.Q
    ST  ChangeBox
    LD  Counter.CV
    ST  CountValue
  END_PROGRAM
```

◇

Übung 13.16 (COUNT85b)

```
  PROGRAM Count85b
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  (*  Hex-Output    byte 1    *)
  (*  Hex-Input     byte 1    *)
  VAR
    Counter : CTD
    Disk      AT %IX0.2: Bool
    ChangeBox AT %QX0.2: Bool
    Start     AT %IX0.1: Bool
    CountValue AT %QB1:  INT
    NumberDisks AT %IB1: INT
  END_VAR
    LD  NumberDisks
    ST  Counter.PV
    LD  Disk
    ST  Counter.CD
    LD  Start
    AND ChangeBox
    ST  Counter.LD
    CAL Counter
    LD  Counter.Q
    ST  ChangeBox
    LD  Counter.CV
    ST  CountValue
  END_PROGRAM
```

◇

Übung 13.17 (MIXER82)

```
Program Mixer82
(*Processes:                 *)
(*  Chemical Process(small) *)
VAR
  V1FF : RS
  V3FF : RS
  Start AT %ix0.7: BOOL
  Stop  AT %ix0.6: BOOL
  LIS1  AT %ix0.1: BOOL
  LIS2  AT %ix0.2: BOOL
  V1    AT %qx0.1: BOOL
  V3    AT %qx0.3: BOOL
  Lamp  AT %qx0.0: BOOL
  Mixer AT %qx0.6: BOOL
  Heater AT %qx0.5: BOOL
  TIC50  AT %ix0.5: BOOL
  V4     AT %qx0.4: BOOL
  LIS3   AT %ix0.3: BOOL
  Run : BOOL
  Fill: BOOL
  Process: BOOL
  Counter: CTD
  Impuls:  TP
  CountValue AT %QB3: SINT
  END_VAR
  ld   3
  st   Counter.PV

  ld   t#5s
  st   Impuls.PT
  cal  Impuls

  ld   Start
  and  Counter.Q
  andn Run
  S    Run
  S    Fill
  st   Counter.LD

  cal  Counter

  ld   Counter.Q
```

```
R    Fill

ld   Fill
andn V3FF.Q1
st   V1FF.S
ld   LIS2
st   V1FF.R1

ld   LIS2
andn V1FF.Q1
st   V3FF.S
ldn  LIS1
st   V3FF.R1
st   Counter.CD

cal  V1FF
cal  V3FF

ld   V1FF.Q1
st   V1

ld   V3FF.Q1
st   V3

ld   Run
andn Fill
s    Process

ld   Process
andn V4
s    Heater

ld   Process
st   Impuls.IN
ld   Impuls.Q
st   Mixer

ld   Process
and  TIC50
r    Heater
s    V4

ldn  LIS3
and  Process
r    run
```

```
    r    Process
    r    V4


    ld   Run
    st   Lamp
    LD  Counter.CV
    int_to_sint
    ST  CountValue

  END_PROGRAM
```

◇

Übung 13.18 (FFFB94)

```
  program FFFB94
  (*Processes:                *)
  (*  Standard-I/O  byte 0    *)
  var
    FF0: FFRS
    FF1: FFSR
    RSet AT %ix0.0: BOOL
    Set  AT %ix0.1: BOOL
    Out0 AT %qx0.0: BOOL
    Out1 AT %qx0.1: BOOL
  end_var

    LD  Set
    ST  FF0.Set
    ST  FF1.Set

    LD  RSet
    ST  FF0.RSet
    ST  FF1.RSet

    CAL FF0
    CAL FF1

    LD  FF0.Y
    ST  Out0
    LD  FF1.Y
    ST  Out1

  end_program
```

◇

Übung 13.19 (GEN92)

```
Program ImpulseGEN92
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Hex-Output    byte 3    *)
(*  -  OR  -                *)
(*  Panel                   *)
var
  Display AT %qb3: SINT
  Counter: CTUD
  Pulse: Puls
  Pulse2:Puls
  runUP AT %i0.7: BOOL
  runDN AT %i0.6: BOOL
end_var
(* Zaehler *)
  ld Pulse.Y
  st Counter.CU
  ld Pulse2.Y
  st Counter.CD

  ld runUP
  st Pulse.run
  ld runDN
  st Pulse2.run

  cal Counter
  cal Pulse
  cal Pulse2

  ld Counter.CV
  int_to_sint
  st Display
end_program
```

◇

Übung 13.20 (TANK101)

```
Program Tank101
(*Processes:                *)
(*  Tanks (small)           *)
(*  Hex-Input       byte 1  *)
var
```

```
    SW1   AT %ix0.0: BOOL
    Start AT %ix0.7: BOOL
    Stop  AT %ix0.6: BOOL
    LIS1  AT %ix0.1: BOOL
    LIS2  AT %ix0.2: BOOL
    V1    AT %qx0.1: BOOL
    V3    AT %qx0.3: BOOL
    V1FF : RS
    V3FF : RS
    Display AT %qb3: sint
    PresetValue AT %ib1: int
    Counter: CTU
    run: bool
  end_var

    LD    Start
    s run
    ld PresetValue
    st Counter.PV

    LD run
    ANDN Counter.Q
    ANDN  V3
    ANDN  LIS2
    ST    V1FF.S
    LD    LIS2
    ST    V1FF.R1
    ST    V3FF.S
    LDN   LIS1
    ST    V3FF.R1

    cal   V1FF
    cal   V3FF

    ld    V3FF.Q1
    st    V3
    st Counter.CU
    ld    V1FF.Q1
    st    V1

    cal  Counter

  (* Ist-Wertausgabe / Ende?*)
    ld Counter.CV
    int_to_sint
```

```
    st Display

    ld Counter.Q
    or Stop
    r run

    ld SW1
    st Counter.R
  end_program
```

◇

Übung 13.21 (7SEG103)
Keine Musterlösung. ◇

Übung 13.22 (FLASH105)

```
  Program Flash105
  (*Processes:                 *)
  (*  Standard-I/O  byte 1     *)
  (*  Standard-I/O  byte 2     *)
  var
    Pulse: TON
    Cycle1: bool
    Bit1 AT %QX1.0: bool
    Bit16 AT %QX2.7: bool
    Byte1 AT %Qw1: uint
    GoLeft AT %IX1.0: BOOL
  end_var
  (* set Bit *)
    ldn Cycle1
    s Cycle1
    s Bit1

  (* Pulsgenerator *)
    ldn Pulse.Q
    st  Pulse.IN
    ld  t#1000ms
    st  Pulse.PT
    cal Pulse

  (* Go!! *)
    ldn Pulse.Q
    jmpc PulseOK
    ld   GoLeft
    jmpcn RunRight
  RunLeft:
```

```
    ld Byte1
    eq 32768
    r Bit16
    r Cycle1

    ld Byte1
    mul 2
    st Byte1
    jmp PulseOK
  RunRight:
    ld Byte1
    div 2
    st Byte1
    eq 0
    s  Bit16
  PulseOK:
  end_program
```

◇

Übung 13.23 (DICE102)

```
Program ProgDice102
(*Processes:                *)
(*  Standard-I/O  byte 0    *)
(*  Hex-Output    byte 0    *)
(*  Standard-I/O  byte 1    *)
(*  Hex-Output    byte 1    *)
 var
  Start0 AT %ix0.0: BOOL
  Start1 AT %ix0.1: BOOL
  OutByte0 AT %qb0: sint
  OutByte1 AT %qb1: sint
  Dice0: Chance16
  Dice1: Chance16
end_var
  ld Start0
  st Dice0.run
  ld Start1
  st Dice1.run

  cal Dice0
  cal Dice1

  ld Dice0.Value
  st OutByte0
  ld Dice1.Value
```

```
  st OutByte1
end_program
```

◇

Übung 13.24 (DICE103)

```
Program ProgDice103
(*Processes:                    *)
(*  Standard-I/O  byte 0    *)
(*  Hex-Output    byte 0    *)
(*  Standard-I/O  byte 1    *)
(*  Hex-Output    byte 1    *)
var
  Start0 AT %ix0.0: BOOL
  Start1 AT %ix0.1: BOOL
  OutByte AT %qb0: byte
  Dice0: Chance16
  Dice1: Chance16
  Dice1Byte: byte
end_var
  ld Start0
  st Dice0.run
  ld Start1
  st Dice1.run

  cal Dice0
  cal Dice1

  ld  Dice1.Value
  mul 16
  sint_to_byte
  st Dice1Byte
  ld Dice0.Value
  sint_to_byte
  or Dice1Byte
  st OutByte
end_program
```

◇

Übung 13.25 (7SEG112)

```
Program SevenSegment112
(*Processes:                    *)
(*  Standard-I/O  byte 2    *)
(*  Hex-Input     byte 2    *)
```

```
(*  7-Segment     byte 1    *)
Var
  InByte1  AT %IB2: byte
  OutByte1 AT %QB1: byte
end_var
  ld InByte1
  byte_to_int
  div 16         (*higher*)
  int_to_7seg
  st OutByte1
end_program
```

◇

Übung 13.26 (7SEG113)

```
Program SevenSegment113
(*Processes:                *)
(*  Standard-I/O  byte 2    *)
(*  Hex-Input     byte 2    *)
(*  7-Segment     byte 0    *)
(*  7-Segment     byte 1    *)
Var
  InByte   AT %IB2: byte
  OutByte0 AT %QB0: byte
  OutByte1 AT %QB1: byte
end_var
  ld InByte
  and 2#00001111     (*lower*)
  byte_to_int
  int_to_7seg
  st OutByte0
  ld InByte
  byte_to_int
  div 16             (*higher*)
  int_to_7seg
  st OutByte1
end_program
```

◇

Übung 13.27 (DICE112)

```
Program ProgDice112
(*Standard-I/O          *)
(*Dice    byte 0        *)
(*Dice    byte 1        *)
var
```

```
    Start0 AT %ix0.0: BOOL
    OutDice0 AT %qb0: BYTE
    Dice0: Chance16
    Start1 AT %ix0.1: BOOL
    OutDice1 AT %qb1: BYTE
    Dice1: Chance16
  end_var
    ld Start0
    st Dice0.run
    cal Dice0
    ld Dice0.Value
    int_to_dice
    st OutDice0

    ld Start1
    st Dice1.run
    cal Dice1
    ld Dice1.Value
    int_to_dice
    st OutDice1
  end_program
```

◇

Übung 13.28 (MIXER123)

```
  Program TankAndMixer123
  (*Processes:                *)
  (*  Chemical Process (small)*)
  VAR
    V1 AT %qx0.1: BOOL
    V3 AT %qx0.3: BOOL
    LIS1 AT %ix0.1: BOOL
    LIS2 AT %ix0.2: BOOL
    Start AT %ix0.7: BOOL
    Mixer AT %qx0.6: BOOL
    Heater AT %qx0.5: BOOL
    TIC50  AT %ix0.5: BOOL
    V2     AT %qx0.2: BOOL
    V4     AT %qx0.4: BOOL
    LIS3   AT %ix0.3: BOOL
    S01:bool
    S02:bool
    S02b:bool
    S03:bool
    S04:bool
```

```
    S05:bool
    S05b:bool
    S06:bool
    S06a:bool
    Cycle1:bool
    TimeMix: TON
    TimeV3: TON
    TimeV4: TON
  END_VAR
    ldn Cycle1
    s   Cycle1
    s   S06a
  (* STEP 1 *)
    LD   S01
    AND  LIS2  (*full?           *)
    R    S01
    S    S02
  (* STEP 2 *)
    LD   S02
    ANDN LIS1  (*empty?          *)
    R    S02
    S    S02b
  (* STEP 2b *)
    LD   S02b
    AND  TimeV3.Q (*Time finished? *)
    R    S02b
    S    S03
  (* STEP 3 *)
    LD   S03
    AND  TIC50 (*Temp. OK? *)
    R    S03
    S    S04
  (* STEP 4 *)
    LD   S04
    AND  TimeMix.Q (*Time finished? *)
    R    S04
    S    S05
  (* STEP 5 *)
    LD   S05
    ANDN LIS3  (*empty?          *)
    R    S05
    S    S05b
  (* STEP 5b *)
    LD   S05b
    AND  TimeV4.Q (*Time finished? *)
```

```
  R    S05b
  S    S06
(* STEP 6 *)
  LD   S06
  ANDN Start (*Start released?*)
  R    S06
  S    S06a
(* STEP 6a *)
  LD   S06a
  AND  Start (*Start pressed? *)
  R    S06a
  S    S01
(* ACTION *)
  LD   S01   (*Step 1 activ?   *)
  ST   V1
  LD   S02   (*Step 2 activ?   *)
  S    V3    (*----- Set V3 ---*)
  LD   S02b   (*Step 2b activ?   *)
  ST   TimeV3.IN
  CAL  TimeV3
  LD   S03   (*Step 3 activ?   *)
  R    V3    (*--- ReSet V3 ---*)
  ST   Heater
  S    Mixer
  S    V2
  LD   S04   (*Step 4 activ?   *)
  ST   TimeMix.IN
  CAL  TimeMix
  LD   S05   (*Step 5 activ?   *)
  S    V4    (*----- Set V4 ---*)
  R    Mixer
  LD   S05b   (*Step 5b activ?   *)
  ST   TimeV4.IN
  CAL  TimeV4
  R    V2
  LD   S06a   (*Step 6a activ?   *)
  R    V4     (*--- ReSet V3 ---*)
(* Timer *)
  LD   t#10s
  ST   TimeMix.PT
  LD   t#2s
  ST   TimeV3.PT
  ST   TimeV4.PT
end_program
```

◇

Übung 13.29 (MIXER124)

```
Program TankAndMixer
(*Processes:                *)
(*  Chemical Process (big)  *)
VAR
  V1 AT %qx0.1: BOOL
  V3 AT %qx0.3: BOOL
  LIS1 AT %ix0.1: BOOL
  LIS2 AT %ix0.2: BOOL
  Start AT %ix0.7: BOOL
  Mixer AT %qx0.6: BOOL
  Heater AT %qx0.5: BOOL
  TIC50  AT %ix0.5: BOOL
  V2     AT %qx0.2: BOOL
  V4     AT %qx0.4: BOOL
  LIS3   AT %ix0.3: BOOL
  LIS11 AT %ix1.1: BOOL
  LIS12 AT %ix1.2: BOOL
  V11 AT %qx1.1: BOOL
  V13 AT %qx1.3: BOOL
  LIS21 AT %ix2.1: BOOL
  LIS22 AT %ix2.2: BOOL
  V21 AT %qx2.1: BOOL
  V23 AT %qx2.3: BOOL
  S01:bool
  S02:bool
  S02b:bool
  S101:bool
  S102:bool
  S102b:bool
  S201:bool
  S202:bool
  S202b:bool
  S03:bool
  S04:bool
  S05:bool
  S05b:bool
  S06:bool
  S06a:bool
  Cycle1:bool
  TimeMix: TON
  TimeV3: TON
  TimeV13: TON
  TimeV23: TON
  TimeV4: TON
```

```
END_VAR
  ldn Cycle1
  s   Cycle1
  s   S06a
(* TANK 0 *)
(* STEP 1 *)
  LD   S01
  AND  LIS2  (*full?          *)
  R    S01
  S    S02
(* STEP 2 *)
  LD   S02
  ANDN LIS1  (*empty?         *)
  R    S02
  S    S02b
(* STEP 2b *)
  LD   S02b
  AND  TimeV3.Q (*Time finished? *)
  R    S02b
  S    S101
(* TANK 1 *)
(* STEP 101 *)
  LD   S101
  AND  LIS12  (*full?          *)
  R    S101
  S    S102
(* STEP 102 *)
  LD   S102
  ANDN LIS11  (*empty?         *)
  R    S102
  S    S102b
(* STEP 102b *)
  LD   S102b
  AND  TimeV13.Q (*Time finished? *)
  R    S102b
  S    S201
(* TANK 2 *)
(* STEP 201 *)
  LD   S201
  AND  LIS22  (*full?          *)
  R    S201
  S    S202
(* STEP 202 *)
  LD   S202
  ANDN LIS21  (*empty?         *)
```

```
    R    S202
    S    S202b
  (* STEP 202b *)
    LD   S202b
    AND  TimeV23.Q (*Time finished? *)
    R    S202b
    S    S03
  (* STEP 3 *)
    LD   S03
    AND  TIC50 (*Temp. OK? *)
    R    S03
    S    S04
  (* STEP 4 *)
    LD   S04
    AND  TimeMix.Q (*Time finished? *)
    R    S04
    S    S05
  (* STEP 5 *)
    LD   S05
    ANDN LIS3  (*empty?          *)
    R    S05
    S    S05b
  (* STEP 5b *)
    LD   S05b
    AND  TimeV4.Q (*Time finished? *)
    R    S05b
    S    S06
  (* STEP 6 *)
    LD   S06
    ANDN Start (*Start released?*)
    R    S06
    S    S06a
  (* STEP 6a *)
    LD   S06a
    AND  Start (*Start pressed? *)
    R    S06a
    S    S01
  (* ACTION *)
    LD   S01   (*Step 1 activ?   *)
    ST   V1
    LD   S02   (*Step 2 activ?   *)
    S    V3    (*------ Set V3 ---*)
    LD   S02b   (*Step 2b activ?   *)
    ST   TimeV3.IN
    CAL  TimeV3
```

```
    LD   S101   (*Step 101 activ?   *)
    ST   V11
    LD   S102   (*Step 102 activ?   *)
    S    V13    (*----- Set V13 ---*)
    LD   S102b   (*Step 102b activ?   *)
    ST   TimeV13.IN
    CAL  TimeV13
    LD   S201   (*Step 201 activ?   *)
    ST   V21
    LD   S202   (*Step 202 activ?   *)
    S    V23    (*----- Set V23 ---*)
    LD   S202b   (*Step 202b activ?   *)
    ST   TimeV23.IN
    CAL  TimeV23

    LD   S03   (*Step 3 activ?   *)
    R    V3    (*--- ReSet V3 ---*)
    R    V13    (*--- ReSet V13 ---*)
    R    V23    (*--- ReSet V23 ---*)
    ST   Heater
    S    Mixer
    S    V2
    LD   S04   (*Step 4 activ?   *)
    ST   TimeMix.IN
    CAL  TimeMix
    LD   S05   (*Step 5 activ?   *)
    S    V4    (*----- Set V4 ---*)
    R    Mixer
    LD   S05b   (*Step 5b activ?   *)
    ST   TimeV4.IN
    CAL  TimeV4
    R    V2
    LD   S06a   (*Step 6a activ?   *)
    R    V4     (*--- ReSet V3 ---*)
  (* Timer *)
    LD   t#10s
    ST   TimeMix.PT
    LD   t#2s
    ST   TimeV3.PT
    ST   TimeV13.PT
    ST   TimeV23.PT
    ST   TimeV4.PT
  end_program
```

◇

Sachverzeichnis